KKT-DPO: DIRECT PREFERENCE OPTIMIZATION WITH OPTIMALITY-DERIVED PER-PROMPT β

Anonymous authors
Paper under double-blind review

ABSTRACT

Direct Preference Optimization (DPO) aligns large language models (LLMs) with human preferences using a coefficient β that controls deviation from a reference model but is fixed across prompts. Existing dynamic methods adjust β across batches, pairs, or prompts using method-specific signals, rather than deriving it from the optimization problem. In this paper, we study a per-prompt KL-constrained formulation, in which each prompt has its own multiplier β_i, and characterize the optimal β_i through the KKT conditions. We show that the optimal multiplier decomposes into the product of the *reward standard deviation* under the reference model and a *dimensionless factor* determined by the standardized reward distribution and the KL budget. When prompts share the same standardized reward distribution, their optimal multipliers are proportional to their reward standard deviations. This suggests that the reward standard deviations determine how the optimal multipliers differ across prompts, and that an anchor sets the overall level. Based on this, we propose **KKT-DPO**, which approximates the relative reward standard deviations using responses sampled from the reference policy and scored by policy-to-reference log-ratios under a lagged policy snapshot. It then uses a coefficient tuned for vanilla DPO as a practical anchor to set the absolute multiplier. Experiments show that KKT-DPO attains the best performance among the baselines on the AlpacaEval and Arena-Hard and improves over DPO on reasoning, mathematics, coding, and instruction-following tasks, demonstrating that our per-prompt coefficients derived from the optimality conditions are effective.

1 INTRODUCTION

Aligning large language models with human preferences is central to building reliable systems (Christiano et al., 2017; Ziegler et al., 2020; Stiennon et al., 2020). Reinforcement learning from human feedback (RLHF) formulates alignment as reward maximization and solves the resulting problem with policy optimization algorithms such as PPO (Ouyang et al., 2022; Schulman et al., 2017). The RLHF objective balances a learned reward against a KL penalty that keeps the policy close to a reference model, where the coefficient β controls the strength of this regularization. Despite its effectiveness, RLHF requires a multi-stage pipeline and costly online optimization. Moreover, optimizing against an imperfect reward model can lead to over-optimization (Gao et al., 2023).

Direct Preference Optimization (DPO) avoids explicit reward-model training and online policy optimization by reparameterizing the optimum of a KL-regularized objective (Rafailov et al., 2023). It trains directly on offline preference pairs through policy-to-reference log-ratios, and has become a widely used approach for preference alignment (Jiang et al., 2024; Dubois et al., 2023).

However, a limitation of vanilla DPO is that it uses a single global coefficient β for all prompts, which can make the KL–reward trade-off too weak for some prompts and too strong for others. Recent work therefore makes the coefficient dynamic. β-DPO adjusts one coefficient per batch using the batch mean of the implicit reward margin and filters examples with the same statistic (Wu et al., 2024). ε-DPO selects among several candidate coefficients for each preference pair according to how a small perturbation of the coefficient changes the preference logit (Lee et al., 2025). MallowsPO rescales the coefficient with a prompt-level dispersion index derived from a Mallows ranking model and estimated from reference-model uncertainty (Chen et al., 2025). More discussion on related work is detailed in Appendix A. These methods show that a dynamic coefficient can be useful,

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.

but they rely on method-specific signals rather than on the optimality conditions of the underlying constrained optimization problem. This raises a more fundamental question:

What determines the theoretically optimal β for each prompt?

In this paper, we study a KL-constrained formulation of preference alignment in which the divergence budget is imposed separately at each prompt, and we interpret β_i as the Lagrange multiplier associated with prompt x_i. Under a common per-prompt budget, the KKT conditions uniquely characterize the optimal multiplier $\beta_i^\star$ for each prompt. We further show that, for this per-prompt problem, $\beta_i^\star$ is invariant to reward translation and positively homogeneous of degree one in the reward, which yields the exact decomposition $\beta_i^\star = \sigma_i \Phi_i$, where σ_i is the reward standard deviation under the reference model and Φ_i is a dimensionless factor that depends only on the standardized reward distribution and the budget. When prompts share the same standardized reward distribution, their optimal multipliers are proportional to their reward standard deviations. The reward standard deviations therefore determine how the optimal multipliers differ across prompts, while a common factor sets the overall level.

Based on this insight, we propose **KKT-DPO**, which turns the decomposition $\beta_i^\star = \sigma_i \Phi_i$ into a per-prompt coefficient that can be computed during training. Since the budget that determines Φ_i is not specified in practice, KKT-DPO sets the overall level by assigning the coefficient β_{anchor} already tuned for vanilla DPO to the prompt with the median reward standard deviation. Under the DPO reward reparameterization, the standard deviation of the policy-to-reference log-ratio over responses sampled from the reference model is proportional to σ_i. We therefore relate σ_i to DPO's implicit reward parameterization without introducing a reward model. In particular, KKT-DPO draws a fixed set of responses from the reference model once before training and scores them under a lagged policy snapshot, which stabilizes both the estimates and training. The resulting per-prompt estimates determine coefficients β_i in KKT-DPO relative to β_{anchor} during training.

Experiments show that KKT-DPO with per-prompt β_i achieves the best performance on AlpacaEval and Arena-Hard benchmarks among the compared baselines, and consistently outperforms DPO on reasoning, mathematics, coding, and instruction-following tasks. Ablations further show that both reference-policy sampling and policy-to-reference scoring are essential to the estimator. Finally, the divergence analysis further shows that the per-prompt coefficients flatten the dependence of realized KL on the estimated reward standard deviation. These results demonstrate that our per-prompt coefficients derived from the optimality conditions are effective.

Our contributions are threefold:

- We identify the DPO coefficient as the Lagrange multiplier of a per-prompt KL constraint, show that the KKT conditions determine it uniquely, and decompose it into the product of the *reward standard deviation* under the reference model and a *dimensionless factor*. Prompts with the same standardized reward distribution therefore carry proportional multipliers (Sections 3.1 and 3.2).
- We propose **KKT-DPO**, which estimates the reward standard deviation from policy-to-reference log-ratios over reference-policy samples scored under a lagged snapshot, and anchors the median prompt at the coefficient already tuned for vanilla DPO (Section 4).
- Experiments show that KKT-DPO achieves the best performance on the open-ended benchmarks among the compared baselines, and outperforms DPO across reasoning, mathematics, coding, and instruction-following tasks. Ablations and divergence analysis further demonstrate that our per-prompt coefficients derived from the optimality conditions are effective (Section 5).

2 PRELIMINARIES

2.1 KL-CONSTRAINED PREFERENCE ALIGNMENT

Let $\mathcal{X}$ and $\mathcal{Y}$ denote the prompt and response spaces, respectively. Let $\mathcal{D}$ be a distribution over prompts, let $\pi(\cdot \mid x)$ be a policy model, let $\pi_{\text{ref}}(\cdot \mid x)$ be the reference model, and let $r : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$

be the reward function. The standard preference alignment maximizes the expected reward under a budget on the average divergence from the reference model,

$$\begin{aligned} \max_{\pi} \quad & \mathbb{E}_{x\sim\mathcal{D}}\mathbb{E}_{y\sim\pi(\cdot|x)}\left[r(x,y)\right] \\ \text{s.t.} \quad & \mathbb{E}_{x\sim\mathcal{D}}\left[D_{\mathrm{KL}}\left(\pi(\cdot\mid x)\,\|\,\pi_{\mathrm{ref}}(\cdot\mid x)\right)\right] \le \overline{K}, \end{aligned} \tag{1}$$

where D_{KL} denotes the KL divergence and $\overline{K}$ is the average KL budget. The Lagrangian relaxation of Eq. 1 introduces a single multiplier $\beta \ge 0$ for the average constraint, and for any $\beta > 0$ maximizing the Lagrangian with respect to π yields the unique optimal policy

$$\pi_\beta(y\mid x) = \frac{\pi_{\mathrm{ref}}(y\mid x)\exp\left(\frac{r(x,y)}{\beta}\right)}{Z_\beta(x)}, \qquad Z_\beta(x) = \sum_{y'\in\mathcal{Y}} \pi_{\mathrm{ref}}(y'\mid x)\exp\left(\frac{r(x,y')}{\beta}\right), \tag{2}$$

where $Z_\beta(x)$ is the normalization constant (Peters et al., 2010; Abdolmaleki et al., 2018; Korbak et al., 2022). The multiplier β controls the strength of the KL penalty that keeps the policy close to the reference model.

2.2 Direct preference optimization

DPO represents the reward implicitly through the policy-to-reference log-ratio induced by Eq. 2 (Rafailov et al., 2023). Inverting that expression gives

$$r(x,y) = \beta\log\frac{\pi_\beta(y\mid x)}{\pi_{\mathrm{ref}}(y\mid x)} + \beta\log Z_\beta(x). \tag{3}$$

Replacing π_β with a parameterized policy π_θ defines the implicit reward $\hat{r}_\theta(x,y) = \beta\log\big(\pi_\theta(y\mid x)/\pi_{\mathrm{ref}}(y\mid x)\big)$, which differs from $r(x,y)$ by the response-independent term $\beta\log Z_\beta(x)$. Under the Bradley–Terry preference model (Bradley & Terry, 1952), preference probabilities depend only on reward differences, and that term therefore cancels between the preferred response y_w and the rejected response y_l. Given a preference dataset $\mathcal{D}_{\mathrm{pref}} = \{(x, y_w, y_l)\}$, DPO minimizes

$$\mathcal{L}_{\mathrm{DPO}}(\theta) = -\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}_{\mathrm{pref}}}\left[\log\sigma\left(\beta\left[\log\frac{\pi_\theta(y_w\mid x)}{\pi_{\mathrm{ref}}(y_w\mid x)} - \log\frac{\pi_\theta(y_l\mid x)}{\pi_{\mathrm{ref}}(y_l\mid x)}\right]\right)\right], \tag{4}$$

where π_θ is the policy being optimized and $\sigma(\cdot)$ denotes the logistic sigmoid. The coefficient β in Eq. 4 therefore plays the role of the KL multiplier associated with the averaged constraint in Eq. 1, and vanilla DPO fixes a single shared value across prompts. In Section 3, we index the prompts as $x_1, \ldots, x_n$, replace the average constraint with a constraint at each prompt, and determine the corresponding optimal per-prompt multipliers.

3 The optimal per-prompt multiplier

In this section, we study a prompt-level KL-constrained view of preference alignment, in which the averaged KL budget in Eq. 1 is replaced by a budget imposed separately at each prompt. The per-prompt problem has a separate Lagrange multiplier β_i for each prompt x_i. Section 3.1 determines the unique optimal multiplier, and Section 3.2 shows that it can be decomposed into the product of the reward standard deviation under the reference model and a dimensionless factor.

3.1 Determination by the KKT conditions

The average constraint in Eq. 1 couples the prompts through a shared divergence budget, increasing divergence at one prompt leaves less budget for the others. We impose the budget at each prompt, which removes the coupling and leaves both the objective and the constraints separable in $\pi(\cdot\mid x_i)$. The problem therefore decomposes into independent per-prompt subproblems.

Assume that $\mathcal{Y}$ is finite, $\pi_{\mathrm{ref}}(\cdot\mid x)$ has full support, and $r(x,\cdot)$ is not constant on that support. For each prompt x_i, consider the per-prompt KL-constrained problem

$$\max_{\pi(\cdot|x_i)} \quad \mathbb{E}_{y\sim\pi(\cdot|x_i)}\left[r(x_i,y)\right] \qquad \text{s.t.} \quad D_{\mathrm{KL}}\left(\pi(\cdot\mid x_i)\,\|\,\pi_{\mathrm{ref}}(\cdot\mid x_i)\right) \le K, \tag{5}$$

where K is the common per-prompt KL budget. Let $\beta_i \geq 0$ be the Lagrange multiplier associated with the KL constraint at x_i, and let λ_i be the multiplier associated with normalization. The Lagrangian is

$$\begin{aligned}\mathcal{L}_i(\pi, \beta_i, \lambda_i) = \mathbb{E}_{y\sim\pi(\cdot|x_i)}\left[r(x_i, y)\right] \\ - \beta_i \left[D_{\mathrm{KL}}\left(\pi(\cdot \mid x_i) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x_i)\right) - K\right] \\ - \lambda_i \left[\sum_{y\in\mathcal{Y}} \pi(y \mid x_i) - 1\right].\end{aligned} \tag{6}$$

For $K > 0$, the feasible set is convex, the objective is linear in $\pi(\cdot \mid x_i)$, and $\pi_{\mathrm{ref}}(\cdot \mid x_i)$ is strictly feasible because its KL divergence to itself is zero. Hence Slater's condition holds, and the KKT conditions are necessary and sufficient for optimality (Boyd & Vandenberghe, 2004).

For $\beta_i > 0$, stationarity of Eq. 6 yields the exponentially tilted policy of Eq. 2 with β_i in place of β, where λ_i is determined by normalization and contributes only the constant $Z_{\beta_i}(x_i)$. We denote this full-support policy by $\pi_{\beta_i}(\cdot \mid x_i)$. After imposing stationarity and normalization, the remaining KKT conditions governing β_i are

$$\begin{aligned}\beta_i &\geq 0 && \text{(dual feasibility)} \\ D_{\mathrm{KL}}\left(\pi_{\beta_i}(\cdot \mid x_i) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x_i)\right) &\leq K && \text{(primal feasibility)} \\ \beta_i \left[D_{\mathrm{KL}}\left(\pi_{\beta_i}(\cdot \mid x_i) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x_i)\right) - K\right] &= 0 && \text{(complementary slackness).}\end{aligned} \tag{7}$$

Let $K_{\max}(x_i)$ denote the minimum divergence from $\pi_{\mathrm{ref}}(\cdot \mid x_i)$ among policies that maximize the expected reward at x_i. The next lemma shows that the KL divergence of the exponentially tilted policy depends on the reward only through its law under the reference model and gives the monotonicity and range properties needed to determine β_i.

Lemma 3.1. *Fix a prompt x, adopt the assumptions of Section 3.1, and let $K_{\max}(x)$ be as defined there.*

- *(i) $D_{\mathrm{KL}}(\pi_\beta(\cdot \mid x) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x))$ depends on the reward only through the distribution of $r(x, y)$ induced by $y \sim \pi_{\mathrm{ref}}(\cdot \mid x)$;*
- *(ii) the divergence is continuous and strictly decreasing in $\beta \in (0, \infty)$, with limits $K_{\max}(x)$ as $\beta \to 0^+$ and 0 as $\beta \to \infty$, and it is therefore a bijection from $(0, \infty)$ onto $(0, K_{\max}(x))$;*
- *(iii) $0 < K_{\max}(x) < \infty$, where positivity follows from the nonconstancy of $r(x, \cdot)$ and finiteness from the full support of $\pi_{\mathrm{ref}}(\cdot \mid x)$.*

The proof is given in Appendix B.1. For $K \geq K_{\max}(x_i)$, a reward-maximizing policy is feasible and the optimal multiplier is zero. At $K = K_{\max}(x_i)$, the minimum-divergence reward-maximizing policy makes the constraint tight while the multiplier remains zero.

We therefore focus on the nontrivial regime $0 < K < K_{\max}(x_i)$. If $\beta_i = 0$, the KL term drops out of Eq. 6, so maximizing the Lagrangian selects only reward-maximizing policies, all of which are infeasible in this regime. Hence $\beta_i^\star > 0$, and complementary slackness in Eq. 7 implies that the KL constraint is active. Lemma 3.1(ii) then shows that this active constraint uniquely determines $\beta_i^\star$.

Proposition 3.2 (Optimal multiplier)**.** *Let $0 < K < K_{\max}(x_i)$. The optimal multiplier $\beta_i^\star$ of Eq. 5 is the unique $\beta > 0$ satisfying*

$$D_{\mathrm{KL}}\left(\pi_\beta(\cdot \mid x_i) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x_i)\right) = K, \tag{8}$$

and the optimal policy at x_i is $\pi_{\beta_i^\star}(\cdot \mid x_i)$.

The proof is given in Appendix B.2. By Lemma 3.1(i), the solution $\beta_i^\star$ depends on the reward only through its distribution under the reference policy, together with the budget. We therefore write $\beta^\star[\cdot]$ for the map from a reward function to this solution, so that $\beta_i^\star = \beta^\star[r(x_i, \cdot)]$.

3.2 Decomposition by positive homogeneity

The policy in Eq. 2 is invariant under the reward translation $r \mapsto r + c$ and under the joint rescaling $(r, \beta) \mapsto (ar, a\beta)$ for any $a > 0$. The divergence at $(ar + c, a\beta)$ therefore equals the divergence at

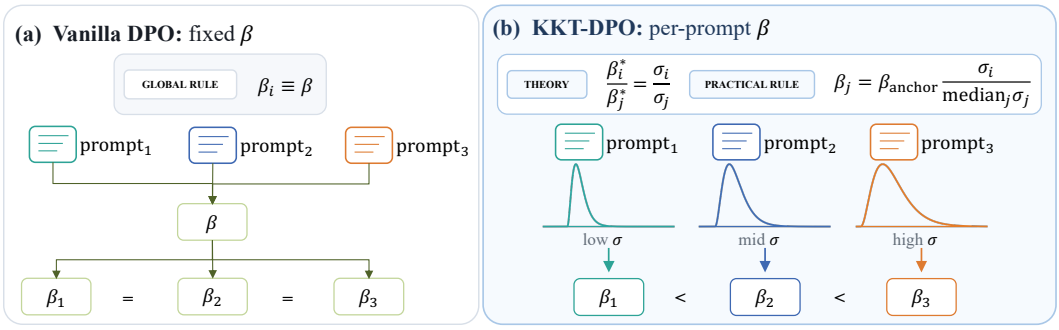


Figure 1: Comparison of KKT-DPO and Vanilla DPO. Vanilla DPO uses a fixed β across prompts, whereas KKT-DPO assigns per-prompt β_i proportional to reward standard deviations.

(r, β), and Lemma 3.1(ii) shows that $K_{\max}(x_i)$ is unchanged as well. Uniqueness in Proposition 3.2 then gives, for every $c \in \mathbb{R}$ and $a > 0$,

$$\beta^\star[r + c] = \beta^\star[r], \qquad \beta^\star[ar] = a\,\beta^\star[r]. \tag{9}$$

The optimal multiplier is thus invariant to reward translation and positively homogeneous of degree one in the reward.

Let μ_i and σ_i denote the mean and standard deviation of $r(x_i, y)$ under $y \sim \pi_{\text{ref}}(\cdot \mid x_i)$, respectively. Full support of the reference model and nonconstancy of the reward imply $\sigma_i > 0$, and the standardized reward is

$$\tilde{r}_i(y) := \frac{r(x_i, y) - \mu_i}{\sigma_i}. \tag{10}$$

Proposition 3.3 (Multiplier decomposition). *Let $0 < K < K_{\max}(x_i)$. Then*

$$\beta_i^\star = \sigma_i\,\Phi_i, \qquad \Phi_i := \beta^\star[\tilde{r}_i]. \tag{11}$$

Proposition 3.3 follows by applying Eq. 9 to $r(x_i, \cdot) = \mu_i + \sigma_i\tilde{r}_i$. Translation invariance gives $\beta_i^\star = \beta^\star[\sigma_i\tilde{r}_i]$, and homogeneity then gives $\beta_i^\star = \sigma_i\beta^\star[\tilde{r}_i] = \sigma_i\Phi_i$. The factor σ_i is the reward standard deviation under the reference model, whereas Φ_i is dimensionless and depends only on the standardized reward distribution and the budget. Since $K_{\max}(x_i)$ is the limit of the divergence as $\beta \to 0^+$, it too depends on the reward only through its law under the reference model, and by the invariance above it is unchanged by translation and rescaling. It therefore depends only on the standardized reward distribution.

Corollary 3.4 (Cross-prompt proportionality). *Let x_i and x_j have the same standardized reward distribution under their respective reference models, that is, let the two reward distributions differ only in location and scale. Then $K_{\max}(x_i) = K_{\max}(x_j)$, and for any budget $0 < K < K_{\max}(x_i)$,*

$$\Phi_i = \Phi_j, \qquad \frac{\beta_i^\star}{\beta_j^\star} = \frac{\sigma_i}{\sigma_j}. \tag{12}$$

Corollary 3.4 gives a sufficient condition under which the reward standard deviations determine the ratios of the optimal multipliers exactly. Appendix D reports an empirical study showing that standardized reward distributions are similar across prompts under the reference policy, which supports treating Φ as approximately shared. The remaining factor Φ is fixed by the budget, which is not specified in practice. Section 4 therefore recovers these ratios from estimated reward standard deviations and sets the overall level by assigning an anchor to the prompt with the median estimate.

4 KKT-DPO

In this section, we turn Proposition 3.3 into a training procedure. Figure 1 contrasts the shared coefficient in vanilla DPO with the per-prompt coefficients in KKT-DPO, which follow the ratios of the reward standard deviations together with an anchor at the median. These ratios are exact under the condition in Corollary 3.4. Section 4.1 fixes the overall level, Section 4.2 expresses the reward standard deviation through policy-to-reference log-ratios, and Section 4.3 estimates it from a lagged policy snapshot and defines the per-prompt objective.

4.1 MEDIAN ANCHORING

Corollary 3.4 determines the ratios of the optimal multipliers but leaves their overall level unspecified. We take the coefficient β_{anchor} already tuned for vanilla DPO as this level and assign it to the prompt with the median reward standard deviation, which gives

$$\beta_i = \beta_{\text{anchor}} \frac{\sigma_i}{\text{median}_j\, \sigma_j}. \tag{13}$$

When all prompts have the same reward standard deviation under the reference model, Eq. 13 assigns β_{anchor} to every prompt and recovers vanilla DPO.

4.2 ESTIMATION FROM POLICY-TO-REFERENCE LOG-RATIOS

Evaluating σ_i requires responses sampled from $\pi_{\text{ref}}(\cdot \mid x_i)$ and scored by the unobserved reward function. We use the implicit reward in Eq. 3 instead. Let π_τ denote the policy in Eq. 2 at coefficient $\tau > 0$. Its implicit reward differs from $r(x_i, \cdot)$ by the response-independent term $\tau \log Z_\tau(x_i)$, which drops out of the standard deviation across responses,

$$\sigma_i = \tau\, \text{sd}_{y \sim \pi_{\text{ref}}(\cdot|x_i)} \left[\log \frac{\pi_\tau(y \mid x_i)}{\pi_{\text{ref}}(y \mid x_i)} \right]. \tag{14}$$

A single τ is shared by all prompts and therefore cancels from the ratio in Eq. 13. The standard deviation of the policy-to-reference log-ratio therefore supplies the quantity required by the allocation.

4.3 THE PER-PROMPT OBJECTIVE

The policy π_τ in Eq. 14 is unavailable because it depends on the unobserved reward function. We approximate its policy-to-reference log-ratios with a lagged policy snapshot $\bar{\pi}$, where a pool $\mathcal{P}$ is the set of prompts covered by the next N training steps. Refreshing $\bar{\pi}$ once per pool keeps the readout aligned with the current policy, and holding it fixed while the pool is scored follows the target-network construction in reinforcement learning (Mnih et al., 2015). For each prompt $x_i \in \mathcal{P}$ we use n responses sampled from the reference model,

$$s_i = \text{sd}_{v=1,\ldots,n} \left[\log \frac{\bar{\pi}(y_v \mid x_i)}{\pi_{\text{ref}}(y_v \mid x_i)} \right], \qquad y_v \sim \pi_{\text{ref}}(\cdot \mid x_i). \tag{15}$$

These responses and their reference log-probabilities are sampled and cached once before training, and the estimates carry no gradient with respect to θ because $\bar{\pi}$ is fixed.

Eq. 15 is subject to sampling variation at finite n. We shrink each estimate toward the pool median,

$$\hat{s}_i = (1 - \alpha) s_i + \alpha\, \text{median}_{j \in \mathcal{P}}\, s_j, \qquad \alpha \in [0, 1). \tag{16}$$

Eq. 16 is an increasing affine map that fixes the pool median, and the median of $\{\hat{s}_i\}_{i \in \mathcal{P}}$ therefore equals the median of $\{s_i\}_{i \in \mathcal{P}}$. The coefficient at each prompt is

$$\beta_i = \beta_{\text{anchor}} \frac{\hat{s}_i}{\text{median}_{j \in \mathcal{P}}\, \hat{s}_j} = \beta_{\text{anchor}} \left[(1 - \alpha) \frac{s_i}{\text{median}_{j \in \mathcal{P}}\, s_j} + \alpha \right]. \tag{17}$$

The medians in Eq. 17 are taken within the pool, and any factor common to the pool therefore cancels. At $\alpha = 0$ the coefficients follow the raw estimates. Increasing α contracts them toward β_{anchor} while the prompt at the pool median keeps β_{anchor}, and all coefficients converge to β_{anchor} as $\alpha \to 1$. A larger estimated reward standard deviation gives a larger coefficient and stronger regularization toward the reference model.

KKT-DPO replaces the global coefficient β in Eq. 4 with the per-prompt coefficient β_i,

$$\mathcal{L}_{\text{KKT-DPO}}(\theta) = -\mathbb{E}_{(x_i, y_w^{(i)}, y_l^{(i)}) \sim \mathcal{D}_{\text{pref}}} \left[\log \sigma \left(\beta_i \left[\log \frac{\pi_\theta(y_w^{(i)} \mid x_i)}{\pi_{\text{ref}}(y_w^{(i)} \mid x_i)} - \log \frac{\pi_\theta(y_l^{(i)} \mid x_i)}{\pi_{\text{ref}}(y_l^{(i)} \mid x_i)} \right] \right) \right]. \tag{18}$$

Training begins with the learning-rate warmup of the standard DPO recipe, which we adopt unchanged. All prompts share β_{anchor} during warmup, and Eq. 18 is then the vanilla DPO objective in Eq. 4. We take the first snapshot at the end of warmup. The policy has moved away from the reference model by then, and every prompt has been trained under the same coefficient. The per-prompt allocation begins with the first pool after warmup. Algorithm C.1 summarizes the complete procedure, and we release our code in the supplementary material.

Table 1: Performance on open-ended evaluation. We report length-controlled (LC) and raw (WR) win rates for AlpacaEval v2.0, and style-controlled (SC) and raw (WR) win rates for Arena-Hard v0.1. Arena-Hard entries include bootstrap 95% confidence intervals as offsets from the point estimate. The best score in each column is in **bold**.

Method	Mistral-Instruct-v0.2 (7B)				Llama-3-Instruct (8B)			
	AlpacaEval v2.0		Arena-Hard		AlpacaEval v2.0		Arena-Hard	
	LC (%)	WR (%)	SC (%)	WR (%)	LC (%)	WR (%)	SC (%)	WR (%)
SFT	23.65	19.38	$12.4^{+1.2}_{-1.2}$	$10.3^{+1.1}_{-1.2}$	30.19	30.67	$15.2^{+1.2}_{-1.3}$	$18.4^{+1.5}_{-1.7}$
DPO	25.48	21.27	$13.7^{+1.4}_{-1.2}$	$11.2^{+1.0}_{-1.1}$	41.78	40.91	$24.8^{+2.1}_{-1.7}$	$28.3^{+2.1}_{-1.9}$
IPO	25.88	20.76	$12.7^{+1.2}_{-1.2}$	$10.6^{+1.1}_{-1.1}$	34.72	35.57	$18.2^{+1.5}_{-1.2}$	$20.9^{+1.5}_{-1.6}$
SimPO	33.05	29.26	$17.1^{+1.4}_{-1.4}$	$15.1^{+1.3}_{-1.3}$	42.58	39.26	$19.5^{+1.5}_{-1.7}$	$24.8^{+2.0}_{-1.7}$
β-DPO	32.16	31.97	$14.7^{+1.3}_{-1.3}$	$14.3^{+1.3}_{-1.2}$	41.52	38.57	$21.8^{+1.6}_{-1.4}$	$25.4^{+1.7}_{-1.8}$
ε-DPO	32.68	27.14	$15.6^{+1.2}_{-1.3}$	$14.1^{+1.4}_{-1.3}$	41.45	40.76	$24.5^{+1.6}_{-1.2}$	$27.3^{+1.7}_{-1.5}$
MallowsPO-θ	31.80	33.04	$14.3^{+1.2}_{-1.1}$	$13.3^{+1.1}_{-1.3}$	42.29	41.12	$25.5^{+1.9}_{-1.4}$	$29.0^{+1.8}_{-1.9}$
KKT-DPO	**34.17**	**34.49**	$\mathbf{18.0}^{+1.7}_{-1.3}$	$\mathbf{18.9}^{+1.5}_{-1.6}$	**44.65**	**43.91**	$\mathbf{27.0}^{+1.8}_{-1.9}$	$\mathbf{30.3}^{+1.8}_{-1.9}$

5 EXPERIMENTS

We evaluate KKT-DPO by training Llama-3-8B-Instruct and Mistral-7B-Instruct-v0.2 in two settings. One targets open-ended benchmarks (Section 5.1), and the other targets reasoning, mathematics, coding, and instruction-following tasks (Section 5.2). In both settings, we use $n = 5$ responses sampled from the reference model, $N = 100$ training steps per pool, and a shrinkage weight of $\alpha = 2/3$ by default. We reproduce all baselines and train KKT-DPO for one epoch on 8 AMD Instinct MI308X GPUs with 192 GB of memory each. Detailed training settings and evaluation protocols are given in Appendix C. The cost of sampling and training is reported in Appendix G.

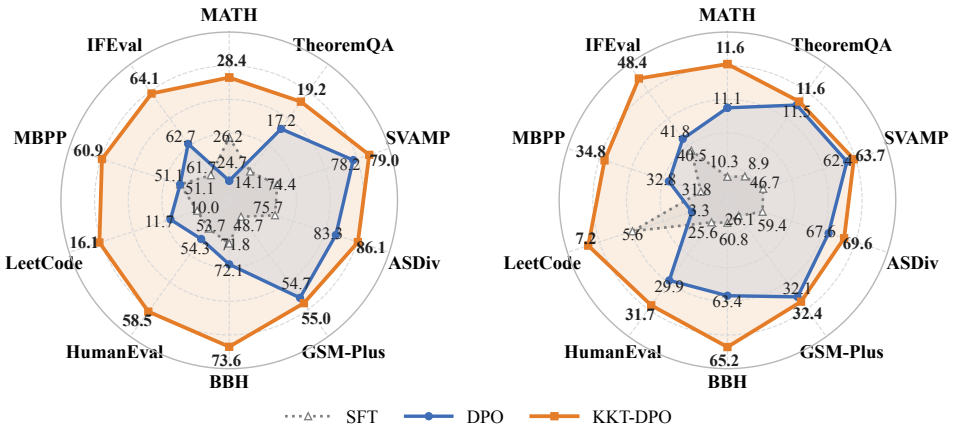
5.1 OPEN-ENDED BENCHMARKS

Setup. We train Llama-3-8B-Instruct on llama3-ultrafeedback and Mistral-7B-Instruct-v0.2 on mistral-instruct-ultrafeedback, following SimPO (Meng et al., 2024). We set $\beta = \beta_{\text{anchor}} = 0.01$. We evaluate on AlpacaEval v2.0 (Dubois et al., 2025) and Arena-Hard v0.1 (Li et al., 2025), reporting the length-controlled and raw win rates (LC, WR) of the former and the style-controlled and raw win rates (SC, WR) of the latter. We compare KKT-DPO against SFT, DPO, IPO (Azar et al., 2024), SimPO (Meng et al., 2024), and the dynamic methods β-DPO (Wu et al., 2024), ε-DPO (Lee et al., 2025), and MallowsPO-θ (Chen et al., 2025).

Results. Table 1 reports the open-ended results. KKT-DPO attains the best score on both models and on both benchmarks. The gains are consistent across both controlled and raw win rates. KKT-DPO gives the best LC and WR on AlpacaEval v2.0 and the best SC and WR on Arena-Hard v0.1 for both backbones. The effect is largest on Mistral, where KKT-DPO improves over DPO by 8.69 points of LC on AlpacaEval v2.0 and 7.70 points of WR on Arena-Hard v0.1. Llama-3 also achieves the best performance on all benchmarks. These results show that per-prompt coefficients derived from the optimality conditions improve open-ended performance across models, supporting the effectiveness of the proposed KKT-DPO. We further report average response lengths and raw win rate standard errors for AlpacaEval v2.0 in Appendix E.

5.2 REASONING, MATHEMATICS, CODING, AND INSTRUCTION-FOLLOWING TASKS

Setup. We train on the mixture of UltraFeedback and UltraInteract using the Eurus setup (Yuan et al., 2025), with $\beta = \beta_{\text{anchor}} = 0.1$. We evaluate on reasoning (BBH (Suzgun et al., 2023)), mathematics (MATH (Hendrycks et al., 2021), TheoremQA (Chen et al., 2023), SVAMP (Patel et al., 2021), ASDiv (Miao et al., 2020), GSM-Plus (Li et al., 2024)), coding (HumanEval (Chen et al., 2021), LeetCode (Guo et al., 2024), MBPP (Austin et al., 2021)), and instruction-following (IFEval (Zhou et al., 2023)) tasks. We compare KKT-DPO against SFT and DPO.



(a) Llama-3-8B-Instruct (b) Mistral-7B-Instruct-v0.2

Figure 2: Reasoning, mathematics, coding, and instruction-following benchmark results. We report Exact Match for BBH, pass@1 for HumanEval and LeetCode, accuracy for MBPP, MATH, TheoremQA, SVAMP, ASDiv, and GSM-Plus. We report prompt-level strict accuracy for IFEval; full results are reported in Table E.3. The best scores are in **bold**.

Table 2: Ablation study. *Sample number* $n = 1$ makes the standard deviation uninformative and recovers vanilla DPO. *Mis-specified sampling* sampled from the other LLM's reference model (Llama-3 $\leftrightarrow$ Mistral) instead of from its own. *Mis-specified scoring* reads $\log \pi_{\text{ref}}$ instead of the policy-to-reference log-ratio. The best score in each column is in **bold**.

Variant	Mistral-Instruct-v0.2 (7B)				Llama-3-Instruct (8B)			
	AlpacaEval v2.0		Arena-Hard		AlpacaEval v2.0		Arena-Hard	
	LC (%)	WR (%)	SC (%)	WR (%)	LC (%)	WR (%)	SC (%)	WR (%)
SFT	23.65	19.38	$12.4^{+1.2}_{-1.2}$	$10.3^{+1.1}_{-1.2}$	30.19	30.67	$15.2^{+1.2}_{-1.3}$	$18.4^{+1.5}_{-1.7}$
sample number $n = 1$	25.39	21.33	$13.6^{+1.4}_{-1.3}$	$11.3^{+0.9}_{-1.1}$	41.69	40.88	$24.6^{+1.1}_{-0.8}$	$28.1^{+1.3}_{-1.1}$
mis-specified sampling	32.77	30.56	$15.8^{+1.5}_{-1.2}$	$14.4^{+1.4}_{-1.2}$	41.19	40.35	$25.2^{+1.2}_{-1.1}$	$28.7^{+1.2}_{-1.2}$
mis-specified scoring	32.99	31.98	$16.2^{+1.4}_{-1.4}$	$14.8^{+1.3}_{-1.2}$	41.32	40.46	$25.5^{+1.2}_{-1.3}$	$28.3^{+1.1}_{-1.3}$
KKT-DPO	**34.17**	**34.49**	$\mathbf{18.0}^{+1.7}_{-1.3}$	$\mathbf{18.9}^{+1.5}_{-1.6}$	**44.65**	**43.91**	$\mathbf{27.0}^{+1.8}_{-1.9}$	$\mathbf{30.3}^{+1.8}_{-1.9}$

Results. Figure 2 shows the results, with full numbers reported in Tables E.1 and E.2. KKT-DPO outperforms DPO on all tasks for both models, raising the average from 51.00 to 54.11 on Llama-3 and from 35.59 to 37.64 on Mistral. On Llama-3 the gain averages 6.16 points across the coding tasks and 1.92 points across the mathematics tasks, with the largest gain of 9.77 points on MBPP. On Mistral the largest gain is 6.66 points on IFEval. Notably, DPO fails to improve on several tasks, falling below SFT on Llama-3's MATH and Mistral's LeetCode, keeping Llama-3's MBPP, and degrading the loose IFEval metrics on Llama-3 (Table E.3). In contrast, KKT-DPO improves MATH on Llama-3 to 28.38, LeetCode on Mistral to 7.22, and MBPP on Llama-3 to 60.90, while also improving the loose IFEval metrics. Since DPO and KKT-DPO share β_{anchor} and every other training setting, these differences are attributable to the per-prompt coefficient allocation in Eq. 17.

5.3 Ablation Study

Table 2 shows the results of ablating Eq. 15. At $n = 1$ the standard deviation is zero, so the implementation falls back to β_{anchor} for every prompt and the objective reduces to vanilla DPO, as stated in Section 4.1. Table 2 shows that the result is close to the DPO result in Table 1. The remaining two variants each change one element of Eq. 15 while keeping training and evaluation fixed. Mis-specified sampling scores responses sampled from the other LLM's reference model instead of from its own. Mis-specified scoring keeps the reference samples but reads $\log \pi_{\text{ref}}$ instead of the policy-to-reference log-ratio. Both substitutions lower every metric on both models, costing

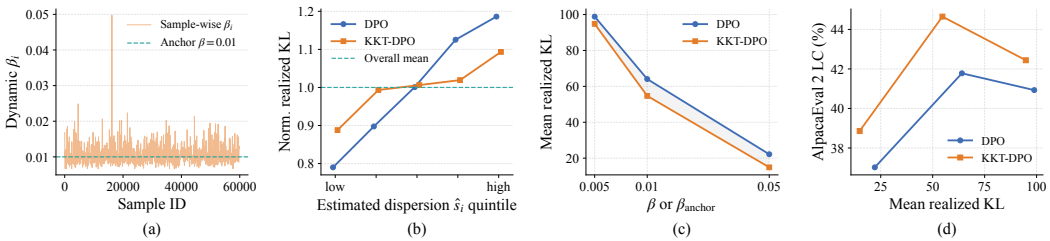


Figure 3: Per-prompt coefficients and realized divergence on Llama-3-8B-Instruct. (a) Per-prompt β_i against prompt index, with the anchor $\beta_{\text{anchor}} = 0.01$ marked. (b) Realized KL by quintile of $\hat{s}_i$, normalized within each method by its own mean so that the panel shows relative allocation rather than overall level. (c) Mean realized KL against β for DPO and against β_{anchor} for KKT-DPO. (d) AlpacaEval v2.0 LC against mean realized KL over the same coefficient sweep as in (c).

3.33–3.46 points of AlpacaEval v2.0 LC on Llama-3 and 1.18–1.40 points on Mistral. These results support the design implied by Eq. 14, which is derived for responses drawn from the reference model and scored by the policy-to-reference log-ratio. We also report sensitivity analyses for the KKT-DPO hyperparameters of α, n, N and β_{anchor} in Appendix F.

5.4 Mechanism analysis

Figure 3 examines the mechanism of KKT-DPO on Llama-3-8B-Instruct. Realized KL is the measured divergence of the trained policy from the reference model and is computed as described in Appendix E.4. Figure 3(a) shows that β_i spans roughly an order of magnitude around $\beta_{\text{anchor}} = 0.01$, with most prompts near the anchor and a right tail reaching several times it. This is the skew that the median in Eq. 13 absorbs. Figure 3(b) shows that under DPO the normalized realized KL rises monotonically with $\hat{s}_i$, so prompts with a larger estimated reward standard deviation absorb disproportionate divergence. Under KKT-DPO, the dependence of realized KL on $\hat{s}_i$ is flatter, and the gap between the highest and lowest quintile narrows by 48.1%. Figure 3(c) shows that the anchor retains global control. Mean realized KL falls monotonically as the coefficient increases for both methods and is lower for KKT-DPO at every value. Figure 3(d) shows that KKT-DPO attains higher AlpacaEval 2 LC at lower realized KL over the same sweep. Together, these panels show that KKT-DPO reallocates divergence across prompts while preserving overall control of the divergence level. At matched coefficients, KKT-DPO gains up to 2.87 LC points while reducing mean realized KL by 4.1% to 32.9%. These effects show that the gain comes from redistributing divergence rather than from spending more of it.

6 Conclusion

In this paper, we study what determines the theoretically optimal coefficient β for each prompt in DPO. We formulate preference alignment as a KL-constrained problem with the budget imposed separately at each prompt, introduce β_i as the Lagrange multiplier of that constraint, and show that the KKT conditions determine the optimal multiplier uniquely. We further show that the optimal multiplier decomposes exactly as $\beta_i^\star = \sigma_i \Phi_i$, where the reward standard deviation under the reference model determines how the optimal multipliers differ across prompts and the dimensionless factor sets their overall level. To turn this decomposition into a practical training rule, we propose **KKT-DPO**. KKT-DPO estimates the relative reward standard deviations from policy-to-reference log-ratios over reference-model samples scored under a lagged policy snapshot, and uses the coefficient already tuned for vanilla DPO as an anchor for the absolute level. Experiments show that KKT-DPO attains the best score on AlpacaEval v2.0 and Arena-Hard, and improves over DPO on reasoning, mathematics, coding, and instruction-following tasks. Ablation and mechanism analyses further indicate that these gains arise from the derived estimator and from reallocating divergence across prompts according to the optimality conditions, rather than from spending more divergence overall. These results show that per-prompt coefficients derived from the optimality conditions provide an effective and principled way to improve direct preference optimization.

AI USE STATEMENT

We used generative AI tools to assist with translation, improve the readability and organization of the manuscript, and provide limited assistance with checking mathematical derivation steps and proof presentation. We did not use generative AI tools to generate data or conduct experiments. All AI-assisted text and derivations were reviewed and verified by the authors, and we take responsibility for the final content of this work, including all text, claims, proofs, and experimental results.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S1ANxQW0b.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. A general theoretical paradigm to understand learning from human preferences. In *International conference on artificial intelligence and statistics*, pp. 4447–4455. PMLR, 2024.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. ISSN 00063444, 14643510. URL http://www.jstor.org/stable/2334029.

Haoxian Chen, Hanyang Zhao, Henry Lam, David Yao, and Wenpin Tang. Mallowspo: Fine-tune your llm with preference dispersions. In *International Conference on Learning Representations*, volume 2025, pp. 31387–31419, 2025.

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, 2021. URL https://arxiv.org/abs/2107.03374.

Wenhu Chen, Ming Yin, Max Ku, Pan Lu, Yixin Wan, Xueguang Ma, Jianyu Xu, Xinyi Wang, and Tony Xia. TheoremQA: A theorem-driven question answering dataset. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7889–7901, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.489. URL https://aclanthology.org/2023.emnlp-main.489/.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Yann Dubois, Chen Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy S Liang, and Tatsunori B Hashimoto. Alpacafarm: A simulation framework for methods that learn from human feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 30039–30069. Curran Associates, Inc., 2023. doi: 10.52202/075280-1308. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/5fc47800ee5b30b8777fdd30abcaaf3b-Paper-Conference.pdf.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled alpacaeval: A simple way to debias automatic evaluators, 2025. URL https://arxiv.org/abs/2404.04475.

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

Alexey Gorbatovski, Boris Shaposhnikov, Alexey Malakhov, Nikita Surnachev, Yaroslav Aksenov, Ian Maksimov, Nikita Balagansky, and Daniil Gavrilov. Learn your reference model for real good alignment. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=H0qIWXXLUR.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. Deepseek-coder: When the large language model meets programming - the rise of code intelligence. *CoRR*, abs/2401.14196, 2024. doi: 10.48550/ARXIV.2401.14196. URL https://doi.org/10.48550/arXiv.2401.14196.

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. *NeurIPS*, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. URL https://arxiv.org/abs/2401.04088.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.792. URL https://aclanthology.org/2023.acl-long.792/.

Tomasz Korbak, Hady Elsahar, Germán Kruszewski, and Marc Dymetman. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. *Advances in Neural Information Processing Systems*, 35:16203–16220, 2022.

Sangkyu Lee, Janghoon Han, Hosung Song, Stanley Jungkyu Choi, Honglak Lee, and Youngjae Yu. KL penalty control via perturbation for direct preference optimization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=00oRAPDWsX.

Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. GSM-plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. 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. 2961–2984, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.163. URL https://aclanthology.org/2024.acl-long.163/.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=KfTf9vFvSn.

Tianlin Liu, Shangmin Guo, Leonardo Bianco, Daniele Calandriello, Quentin Berthet, Felipe Llinares, Jessica Hoffmann, Lucas Dixon, Michal Valko, and Mathieu Blondel. Decoding-time realignment of language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=5PAF7PAY2Y.

Yu Meng, Mengzhou Xia, and Danqi Chen. Simpo: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*, 37:124198–124235, 2024.

Shen-yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing English math word problem solvers. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 975–984, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.92. URL https://aclanthology.org/2020.acl-main.92/.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple math word problems? In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tur, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou (eds.), *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.naacl-main.168. URL https://aclanthology.org/2021.naacl-main.168/.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2021. URL https://openreview.net/forum?id=ToWi1RjuEr8.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

Jan Peters, Katharina Mulling, and Yasemin Altun. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pp. 1607–1612, 2010.

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

Rafael Rafailov, Yaswanth Chittepu, Ryan Park, Harshit Sikchi, Joey Hejna, W. Bradley Knox, Chelsea Finn, and Scott Niekum. Scaling laws for reward model overoptimization in direct alignment algorithms. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 126207–126242. Curran Associates, Inc., 2024. doi: 10.52202/079017-4009. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/e45caa3d5273d105b8d045e748636957-Paper-Conference.pdf.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017. URL http://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, 2024. URL https://arxiv.org/abs/2402.03300.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 3008–3021. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1f89885d556929e98d3ef9b86448f951-Paper.pdf.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Junkang Wu, Yuexiang Xie, Zhengyi Yang, Jiancan Wu, Jinyang Gao, Bolin Ding, Xiang Wang, and Xiangnan He. β-dpo: Direct preference optimization with dynamic β. *Advances in Neural Information Processing Systems*, 37:129944–129966, 2024.

Junkang Wu, Xue Wang, Zhengyi Yang, Jiancan Wu, Jinyang Gao, Bolin Ding, Xiang Wang, and Xiangnan He. AlphaDPO: Adaptive reward margin for direct preference optimization. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=ETLKYVMVLt.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Boji Shan, Zeyuan Liu, Jia Deng, Huimin Chen, Ruobing Xie, et al. Advancing llm reasoning generalists with preference trees. In *International Conference on Learning Representations*, volume 2025, pp. 24897–24919, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2020. URL https://arxiv.org/abs/1909.08593.

A Related Work

Direct alignment algorithms. Reinforcement learning from human feedback fits a reward model to preferences and optimizes it under a KL penalty, commonly with PPO (Christiano et al., 2017; Ziegler et al., 2020; Stiennon et al., 2020; Ouyang et al., 2022; Schulman et al., 2017). DPO reparameterizes the optimum of the KL-regularized objective into a preference model and removes explicit reward-model training and online rollouts (Rafailov et al., 2023). IPO replaces the logistic loss with a squared loss (Azar et al., 2024), whereas SimPO uses a reference-free implicit reward and a fixed target margin (Meng et al., 2024). These methods use one global objective coefficient across the training corpus.

Dynamic coefficients in preference optimization. Several methods vary an objective coefficient across batches or examples. β-DPO compares the batch mean implicit-reward margin with a moving baseline, assigns one coefficient to the batch, and filters examples using the same margin (Wu et al., 2024). ε-DPO selects among $\beta/(1+\varepsilon)$, β, and $\beta/(1-\varepsilon)$ for each preference pair according to a local perturbation criterion (Lee et al., 2025). MallowsPO assigns a prompt-dependent factor derived from a Mallows ranking model and estimates it using reference-model uncertainty (Chen et al., 2025). Alpha-DPO adapts the reward margin rather than the coefficient (Wu et al., 2025), and decoding-time realignment varies the coefficient at inference rather than during training (Liu et al., 2024). Among methods that vary the training-time coefficient, the signal is method-specific: for example, a batch statistic, a local perturbation test, or a ranking-model assumption. We instead ask which allocation follows from the constraint itself. Section 3.1 identifies this family of objectives

as the Lagrangian of the KL constraint imposed separately at each prompt, so its coefficients are the dual variables of those constraints; Proposition 3.2 selects the member of the family whose constraint is active at every prompt, and Proposition 3.3 shows that this member decomposes into the reward standard deviation under the reference model and a dimensionless factor determined by the standardized reward distribution and the budget.

KL-constrained policy optimization. The exponentially tilted policy in Eq. 2 is the solution of KL-regularized policy improvement and appears in reward-weighted regression, advantage-weighted regression, REPS, and MPO (Peters & Schaal, 2007; Peng et al., 2021; Peters et al., 2010; Abdolmaleki et al., 2018). The same objective can also be interpreted as distribution matching (Korbak et al., 2022). REPS and MPO already obtain the multiplier from the constraint rather than fixing it in advance, but the constraint they impose is a divergence averaged over states and therefore yields one multiplier for the aggregate constraint; Section 3.1 imposes the constraint at each prompt and obtains one multiplier per prompt. Temporal adaptations are complementary: RLHF can update a global penalty to track a target divergence (Ziegler et al., 2020), while TR-DPO periodically changes the reference model (Gorbatovski et al., 2025). Because quality need not vary monotonically with realized divergence (Gao et al., 2023; Rafailov et al., 2024), Section 5.4 reports quality against the realized divergence rather than at a single coefficient.

Reward standardization. GRPO standardizes rewards or advantages within a group of responses to the same prompt (Shao et al., 2024), and subsequent work studies the optimization consequences of that normalization (Liu et al., 2025). The quantity it forms is the standardized reward $\tilde{r}_i$ of Section 3.2. Proposition 3.3 assigns that standardization a further role: scaling the reward by its dispersion and scaling the coefficient by its dispersion are two sides of one identity. GRPO acts on the reward side, whereas Section 4.1 acts on the coefficient side and therefore preserves the DPO functional form while changing its per-prompt coefficient.

B Proofs for Section 3

B.1 Proof of Lemma 3.1

Throughout this section, the response space $\mathcal{Y}$ is finite, $\pi_{\mathrm{ref}}(\cdot \mid x)$ has full support, and $r(x, \cdot)$ is nonconstant. Fix a prompt x and suppress it from the notation. Let

$$q(y) = \pi_{\mathrm{ref}}(y \mid x), \qquad r(y) = r(x, y). \tag{19}$$

For $t \geq 0$, define the cumulant-generating function

$$\Psi(t) = \log \mathbb{E}_{y \sim q}\left[e^{tr(y)}\right] = \log \sum_{y \in \mathcal{Y}} q(y) e^{tr(y)}. \tag{20}$$

and define the exponentially tilted distribution

$$p_t(y) = q(y) \exp\left(tr(y) - \Psi(t)\right). \tag{21}$$

The policy in Eq. 2 is p_t with $t = 1/\beta$.

Let

$$r^\star = \max_{y \in \mathcal{Y}} r(y), \qquad \mathcal{Y}^\star = \{y \in \mathcal{Y} : r(y) = r^\star\}, \qquad Q^\star = q(\mathcal{Y}^\star), \tag{22}$$

and define

$$K_{\max}(x) = -\log Q^\star. \tag{23}$$

Lemma B.1. *Fix a prompt x, adopt the assumptions of Section 3.1, and let $K_{\max}(x)$ be as defined there.*

(i) $D_{\mathrm{KL}}(\pi_\beta(\cdot \mid x) \,\|\, \pi_{\mathrm{ref}}(\cdot \mid x))$ depends on the reward only through the distribution of $r(x, y)$ induced by $y \sim \pi_{\mathrm{ref}}(\cdot \mid x)$;

(ii) the divergence is continuous and strictly decreasing in $\beta \in (0, \infty)$, with limits $K_{\max}(x)$ as $\beta \to 0^+$ and 0 as $\beta \to \infty$, and it is therefore a bijection from $(0, \infty)$ onto $(0, K_{\max}(x))$;

(iii) $0 < K_{\max}(x) < \infty$, *where positivity follows from the nonconstancy of* $r(x, \cdot)$ *and finiteness from the full support of* $\pi_{\mathrm{ref}}(\cdot \mid x)$.

Proof. Because $\mathcal{Y}$ is finite, Ψ is differentiable to every order. Differentiating Eq. 20 gives

$$\Psi'(t) = \mathbb{E}_{p_t}[r], \qquad \Psi''(t) = \mathrm{Var}_{p_t}[r]. \tag{24}$$

Since q has full support, so does p_t for every finite t. The reward is nonconstant on this support, and therefore

$$\Psi''(t) = \mathrm{Var}_{p_t}[r] > 0. \tag{25}$$

Equation 21 gives

$$\log \frac{p_t(y)}{q(y)} = tr(y) - \Psi(t). \tag{26}$$

Taking the expectation under p_t yields

$$D_{\mathrm{KL}}(p_t \| q) = t\Psi'(t) - \Psi(t). \tag{27}$$

Thus the divergence depends on r only through the distribution of $r(y)$ under $y \sim q$, equivalently through its cumulant-generating function Ψ, proving part (i).

Differentiating Eq. 27 with respect to t gives

$$\frac{\mathrm{d}}{\mathrm{d}t} D_{\mathrm{KL}}(p_t \| q) = t\Psi''(t) = t\,\mathrm{Var}_{p_t}[r] > 0 \qquad (t > 0). \tag{28}$$

Using $t = 1/\beta$ and the chain rule,

$$\frac{\mathrm{d}}{\mathrm{d}\beta} D_{\mathrm{KL}}(\pi_\beta \| q) = -\frac{\mathrm{Var}_{\pi_\beta}[r]}{\beta^3} < 0. \tag{29}$$

This proves strict monotonicity in the multiplier.

As $\beta \to \infty$, $t \to 0$ and $p_t \to q$. Because $\Psi(0) = 0$, Eq. 27 gives

$$\lim_{\beta \to \infty} D_{\mathrm{KL}}(\pi_\beta \| q) = 0. \tag{30}$$

For the other limit, Eq. 21 can be written as

$$p_t(y) = \frac{q(y) e^{-t(r^\star - r(y))}}{Q^\star + \sum_{z \notin \mathcal{Y}^\star} q(z) e^{-t(r^\star - r(z))}}. \tag{31}$$

Every $z \notin \mathcal{Y}^\star$ has a strictly positive reward gap. Hence, as $t \to \infty$,

$$p_t(y) \longrightarrow q^\star(y) = \begin{cases} q(y)/Q^\star, & y \in \mathcal{Y}^\star, \\ 0, & y \notin \mathcal{Y}^\star. \end{cases} \tag{32}$$

Since $\mathcal{Y}$ is finite and $p_t(y) \to q^\star(y)$ for every y, the KL sum converges termwise, with the convention $0 \log 0 = 0$. Therefore

$$\lim_{\beta \to 0^+} D_{\mathrm{KL}}(\pi_\beta \| q) = D_{\mathrm{KL}}(q^\star \| q) = -\log Q^\star = K_{\max}(x). \tag{33}$$

Together with Eq. 29, these limits prove part (ii).

A policy p maximizes expected reward if and only if it is supported on $\mathcal{Y}^\star$. For every such policy,

$$\begin{aligned} D_{\mathrm{KL}}(p \| q) &= \sum_{y \in \mathcal{Y}^\star} p(y) \log \frac{p(y)}{q(y)} \\ &= D_{\mathrm{KL}}(p \| q^\star) - \log Q^\star \\ &\geq -\log Q^\star = K_{\max}(x), \end{aligned} \tag{34}$$

with equality if and only if $p = q^\star$. Nonconstancy of the reward and full support of q imply $\mathcal{Y}^\star \mathcal{Y}$, so $Q^\star < 1$ and $K_{\max}(x) > 0$. This proves part (iii). □

B.2 Proof of Proposition 3.2

Proof. Fix x_i, abbreviate $q(y) = \pi_{\mathrm{ref}}(y \mid x_i)$ and $r(y) = r(x_i, y)$, and suppose that $0 < K < K_{\max}(x_i)$. Lemma 3.1 gives a unique $\widehat{\beta} > 0$ satisfying

$$D_{\mathrm{KL}}\left(\pi_{\widehat{\beta}}(\cdot \mid x_i) \,\middle\|\, \pi_{\mathrm{ref}}(\cdot \mid x_i)\right) = K. \tag{35}$$

Let

$$\widehat{p}(y) = \frac{q(y)e^{r(y)/\widehat{\beta}}}{Z_{\widehat{\beta}}(x_i)}. \tag{36}$$

The policy $\widehat{p}$ is strictly positive and normalized. Stationarity of the Lagrangian in Eq. 6 requires

$$r(y) - \widehat{\beta}\left(\log\frac{\widehat{p}(y)}{q(y)} + 1\right) - \widehat{\lambda} = 0. \tag{37}$$

Because

$$\log\frac{\widehat{p}(y)}{q(y)} = \frac{r(y)}{\widehat{\beta}} - \log Z_{\widehat{\beta}}(x_i), \tag{38}$$

Eq. 37 holds with

$$\widehat{\lambda} = \widehat{\beta}\left(\log Z_{\widehat{\beta}}(x_i) - 1\right). \tag{39}$$

Equation 35 gives primal feasibility and activity of the KL constraint; $\widehat{\beta} > 0$ gives dual feasibility; and

$$\widehat{\beta}\left[D_{\mathrm{KL}}(\widehat{p}\|q) - K\right] = 0 \tag{40}$$

gives complementary slackness.

The sufficiency of these conditions follows directly from the Gibbs identity. For any policy p on $\mathcal{Y}$,

$$\begin{aligned}\mathbb{E}_p[r] - \widehat{\beta}\left[D_{\mathrm{KL}}(p\|q) - K\right] \\ = \widehat{\beta}\left[K + \log Z_{\widehat{\beta}}(x_i)\right] - \widehat{\beta}D_{\mathrm{KL}}(p\|\widehat{p}).\end{aligned} \tag{41}$$

Thus $\widehat{p}$ uniquely maximizes the Lagrangian over the probability simplex. For every feasible p,

$$\begin{aligned}\mathbb{E}_{\widehat{p}}[r] &= \mathcal{L}_i(\widehat{p}, \widehat{\beta}, \widehat{\lambda}) \\ &\geq \mathcal{L}_i(p, \widehat{\beta}, \widehat{\lambda}) \\ &= \mathbb{E}_p[r] + \widehat{\beta}\left[K - D_{\mathrm{KL}}(p\|q)\right] \\ &\geq \mathbb{E}_p[r].\end{aligned} \tag{42}$$

Therefore $\widehat{p}$ solves Eq. 5.

It remains to establish uniqueness of the multiplier. The dual function for $\beta > 0$ is

$$g(\beta) = \sup_p \left\{\mathbb{E}_p[r] - \beta\left[D_{\mathrm{KL}}(p\|q) - K\right]\right\} = \beta K + \beta\Psi\left(\frac{1}{\beta}\right). \tag{43}$$

Differentiating and using Eq. 27 yields

$$g'(\beta) = K - D_{\mathrm{KL}}(\pi_\beta\|q), \qquad g''(\beta) = \frac{\mathrm{Var}_{\pi_\beta}[r]}{\beta^3} > 0. \tag{44}$$

Hence g is strictly convex. Lemma 3.1 gives

$$\lim_{\beta\to 0^+} g'(\beta) = K - K_{\max}(x_i) < 0, \qquad \lim_{\beta\to\infty} g'(\beta) = K > 0. \tag{45}$$

The unique minimizer is therefore the unique root in Eq. 35. A zero multiplier cannot be optimal because it selects a reward-maximizing policy, and every such policy has divergence at least $K_{\max}(x_i) > K$.

Finally, the reference model is strictly feasible because its divergence is zero. Slater's condition therefore guarantees KKT necessity and strong duality. Consequently, $\beta_i^\star = \widehat{\beta}$, and the optimal policy is $\pi_{\beta_i^\star}(\cdot \mid x_i)$. □

B.3 Proof of Proposition 3.3

Proof. Fix the reference model and budget, and write $\beta^\star[r]$ for the unique multiplier characterized by Proposition 3.2. For any $c \in \mathbb{R}$, $a > 0$, and $\beta > 0$, exponential tilting gives

$$\pi_\beta^{r+c} = \pi_\beta^r, \qquad \pi_{a\beta}^{ar} = \pi_\beta^r, \tag{46}$$

because response-independent factors cancel from the normalizer. Translation and positive rescaling also leave the reward-maximizing set, and therefore $K_{\max}$, unchanged. Uniqueness in Proposition 3.2 implies

$$\beta^\star[r + c] = \beta^\star[r], \qquad \beta^\star[ar] = a\beta^\star[r]. \tag{47}$$

For prompt x_i, full support and reward nonconstancy imply $\sigma_i > 0$, and

$$r(x_i, \cdot) = \mu_i + \sigma_i \tilde{r}_i. \tag{48}$$

Applying Eq. 47 first to the translation and then to the positive rescaling gives

$$\beta_i^\star = \beta^\star[\mu_i + \sigma_i \tilde{r}_i] = \sigma_i \beta^\star[\tilde{r}_i] = \sigma_i \Phi_i. \tag{49}$$

Thus Φ_i depends only on the standardized reward distribution and the budget. □

B.4 Proof of Corollary 3.4

Proof. If x_i and x_j have the same standardized reward distribution under their respective reference policies, then the cumulant-generating functions of their standardized rewards coincide. By Lemma 3.1(i), their KL-divergence curves coincide as functions of the multiplier. Taking the limit as $\beta \to 0^+$ and applying Lemma 3.1(ii) gives

$$K_{\max}(x_i) = K_{\max}(x_j). \tag{50}$$

For any shared budget $0 < K < K_{\max}(x_i)$, uniqueness in Proposition 3.2 therefore gives $\Phi_i = \Phi_j$. Proposition 3.3 then yields

$$\frac{\beta_i^\star}{\beta_j^\star} = \frac{\sigma_i \Phi_i}{\sigma_j \Phi_j} = \frac{\sigma_i}{\sigma_j}. \tag{51}$$

□

C Training and Evaluation Details

C.1 Models, training data, and training configurations

We use Llama-3-8B-Instruct[1] and Mistral-7B-Instruct-v0.2[2] in all experiments. All methods are trained for one epoch with an effective batch size of 128, and we select the last checkpoint rather than a validation-selected best checkpoint.

Open-ended experiments follow the SimPO training setup and use the corresponding on-policy UltraFeedback datasets released with SimPO (Meng et al., 2024): `princeton-nlp/llama3-ultrafeedback`[3] and `princeton-nlp/mistral-instruct-ultrafeedback`[4]. Each prompt already includes five responses together with the preferred and rejected response selected by the released dataset. The dataset construction uses PairRM to choose the highest- and lowest-scoring responses (Jiang et al., 2023); we use the released pairs directly and do not rerun PairRM.

Reasoning, mathematics, coding, and instruction-following experiments follow the training data construction used in Yuan et al. (2025), *Advancing LLM Reasoning Generalists with Preference Trees*. Concretely, we follow the Eurus repository and train on the mixture of `UltraInteract_pair`[5] and UltraFeedback. The resulting corpus contains approximately 275K

[1] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct
[2] https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2
[3] https://huggingface.co/datasets/princeton-nlp/llama3-ultrafeedback
[4] https://huggingface.co/datasets/princeton-nlp/mistral-instruct-ultrafeedback
[5] https://huggingface.co/datasets/openbmb/UltraInteract_pair

Table C.1: Open-ended training configurations.

Configuration	Mistral-Instruct-v0.2	Llama-3-Instruct
Model	Mistral-7B-Instruct-v0.2	Llama-3-8B-Instruct
Training data	On-policy UltraFeedback	On-policy UltraFeedback
β_{anchor}	0.01	0.01
Maximum sequence length	2048	2048
Overlength handling	Filter	Filter
Epochs	1	1
Effective batch size	128	128
Learning rate	5×10^{-7}	7×10^{-7}
Checkpoint	Last	Last
KKT-DPO defaults	$n = 5$, $N = 100$ steps, $\alpha = 2/3$	
Hardware	8 AMD Instinct MI308X GPUs (192 GB each)	

Table C.2: Training configurations for reasoning, mathematics, coding, and instruction-following tasks.

Configuration	Mistral-Instruct-v0.2	Llama-3-Instruct
Model	Mistral-7B-Instruct-v0.2	Llama-3-8B-Instruct
Training data	`UltraInteract_pair` + UltraFeedback	
β_{anchor}	0.1	0.1
Maximum sequence length	4096	4096
Overlength handling	Filter	Filter
Epochs	1	1
Effective batch size	128	128
Learning rate	5×10^{-7}	7×10^{-7}
Checkpoint	Last	Last
KKT-DPO defaults	$n = 5$, $N = 100$ steps, $\alpha = 2/3$	
Hardware	8 AMD Instinct MI308X GPUs (192 GB each)	

preference pairs in a 79:21 ratio. Examples exceeding the maximum sequence length are filtered rather than truncated.

For all baselines, we follow the released implementations as closely as possible. The open-ended training configuration, including learning rates, follows the SimPO codebase[6]. For the dynamic baselines, we use the released implementations of β-DPO from [7]; ε-DPO from [8]; and MallowsPO from [9]. IPO uses the native `DPOTrainer` in Hugging Face TRL 0.13.0 with `loss_type="ipo"`. In this implementation, the loss includes per-token normalization, which follows TRL and is not part of the original IPO objective. For MallowsPO, we report only the MallowsPO-θ variant implemented in the released repository; the unimplemented MallowsPO-ϕ branch is not included in our comparison.

For the reasoning, mathematics, coding, and instruction-following setting, we compare against SFT and DPO only. Extending the comparison to existing dynamic β baselines would require method-specific retuning—and, for MallowsPO-θ, additional token-level entropy preprocessing—on the substantially larger 275K-pair, 4096-token training setup. In particular, `UltraInteract_pair` contains multi-turn conversations, and directly transferring open-ended settings would not provide a reliable comparison. When equal and sufficient tuning budgets cannot be provided for all methods, reporting untuned downstream results would introduce a systematic unfairness. We therefore restrict the comparison to baselines for which we can ensure a reliable and matched training protocol.

Table C.1 and Table C.2 summarize the training configurations.

[6] https://github.com/princeton-nlp/SimPO
[7] https://github.com/junkangwu/beta-DPO
[8] https://github.com/oddqueue/e-dpo
[9] https://github.com/haoxian-chen/MallowsPO

C.2 KKT-DPO IMPLEMENTATION

The reference model is frozen throughout training. For each training prompt x_i, we draw $n = 5$ responses from the reference model and cache their response tokens together with the corresponding reference log probabilities. Every $N = 100$ optimizer steps, a lagged policy snapshot $\bar{\pi}$ scores the cached responses needed by the next training window without gradients. For prompt x_i, the implementation computes

$$s_i = \mathrm{sd}_{v=1,\ldots,n} \left[\log \bar{\pi}(y_v \mid x_i) - \log \pi_{\mathrm{ref}}(y_v \mid x_i) \right], \tag{52}$$

where $y_v \sim \pi_{\mathrm{ref}}(\cdot \mid x_i)$. This is the implementation form of Eq. 15.

Let m be the median of s_i within the current pool. The shrunk estimate and coefficient are

$$\hat{s}_i = (1 - \alpha) s_i + \alpha m, \qquad \beta_i = \beta_{\mathrm{anchor}} \frac{\hat{s}_i}{m}, \tag{53}$$

with $\alpha = 2/3$. Because this shrinkage is an increasing affine map that fixes the pool median, Eq. 53 is equivalent to Eq. 17. As described in Section 4.3, training begins with the standard DPO warmup under the anchor β_{anchor}, and the first lagged snapshot $\bar{\pi}$ is taken at the end of warmup. The first pool after warmup is therefore scored with that warmup-completed snapshot rather than with a static fallback. Only a pool whose median is undefined, zero, or non-finite falls back to the static anchor. During each gradient update, β_i is treated as detached, so the backward pass is exactly the standard DPO backward pass with a per-example coefficient lookup.

For a fixed cache of $\hat{s}_i$ values and a fixed pool median m, changing β_{anchor} scales all implementation coefficients uniformly. Consequently, $\beta_i / \beta_{\mathrm{anchor}}$ and the coefficient ranking are invariant to the anchor scale. This is an analytic property of Eq. 53 rather than a separate empirical claim.

Additional hyperparameter sensitivity analyses, including variations of the KKT-DPO implementation choices, are reported in Section F.

C.3 EVALUATION PROTOCOLS

Open-ended evaluation. AlpacaEval v2.0 uses the official evaluation code and default configuration [10] and reports length-controlled win rate (LC) and raw win rate (WR) over 805 prompts (Dubois et al., 2025). We use `gpt-4o` as the judge model for all methods.

Arena-Hard v0.1 uses the official evaluation code and default configuration [11] on 500 prompts and reports style-controlled win rate (SC) and raw win rate (WR) (Li et al., 2025). We use `gpt-4.1` as the judge model for all methods.

Because our AlpacaEval and Arena-Hard runs use judge models that differ from some public leaderboard settings, the absolute numbers are not expected to match those leaderboards exactly. All methods in our comparison, however, are evaluated under the same judge model within each benchmark.

Reasoning, mathematics, coding, and instruction-following tasks evaluation. These benchmarks use the unmodified default evaluation pipeline released with Yuan et al. (2025), *Advancing LLM Reasoning Generalists with Preference Trees* [12]. Both generation and benchmark-specific scoring follow that repository directly. Table C.3 reports the evaluation set sizes, metrics, and decoding settings.

IFEval contains 541 prompts and 834 verifiable instructions (Zhou et al., 2023). Prompt-level evaluation requires every instruction in a prompt to pass, whereas instruction-level evaluation scores each instruction independently. Strict scoring evaluates the original response, and loose scoring evaluates the standardized response transformations defined by the official implementation. Generation uses the model chat template without an added system prompt.

[10] https://github.com/tatsu-lab/alpaca_eval

[11] https://github.com/lmarena/arena-hard-auto

[12] https://github.com/OpenBMB/Eurus

Table C.3: Reasoning, mathematics, coding, and instruction-following tasks evaluation settings. Each entry uses one generated response per prompt.

Category	Benchmark	Examples	Metric	Decoding
Mathematics	MATH	5,000	Accuracy	Greedy, max 1,024
	TheoremQA	800	Accuracy	Greedy, max 1,024
	SVAMP	1,000	Accuracy	Greedy, max 1,024
	ASDiv	618	Accuracy	Greedy, max 1,024
	GSM-Plus	10,552	Accuracy	Greedy, max 1,024
Coding	HumanEval	164	pass@1	Temperature 0.2, max 1,024
	LeetCode	180	pass@1	Greedy, max 1,024
	MBPP	399	Execution accuracy	Greedy, max 512
Reasoning	BBH (CoT)	6,511	Exact match	Greedy, max 2,048
instruction-following	IFEval	541 prompts	Strict/loose	Temperature 0.7, max 1,024

Algorithm C.1 KKT-DPO

Require: preference data $\{(x_i, y_w^{(i)}, y_l^{(i)})\}$, reference model π_{ref}, anchor β_{anchor}, warmup steps W, steps per pool N, reference sample count n, shrinkage weight α, learning rate η

Draw $y_1, \dots, y_n \sim \pi_{\text{ref}}(\cdot \mid x_i)$ for each prompt x_i and cache $\log \pi_{\text{ref}}(y_v \mid x_i)$
Initialize θ from π_{ref}
Run W warmup steps with $\beta_i = \beta_{\text{anchor}}$ for all i ▷ vanilla DPO
for $k = 0, 1, 2, \dots$ **do**
  $\mathcal{P}_k \leftarrow$ prompts covered by the next N training steps
  $\bar{\pi} \leftarrow \pi_\theta$ ▷ fixed while $\mathcal{P}_k$ is scored
  $s_i \leftarrow \text{sd}_{v=1,\dots,n} [\log \bar{\pi}(y_v \mid x_i) - \log \pi_{\text{ref}}(y_v \mid x_i)]$ for all $i \in \mathcal{P}_k$
  $\hat{s}_i \leftarrow (1-\alpha) s_i + \alpha \,\text{median}_{j \in \mathcal{P}_k} s_j$ for all $i \in \mathcal{P}_k$
  $\beta_i \leftarrow \beta_{\text{anchor}} \hat{s}_i / \,\text{median}_{j \in \mathcal{P}_k} \hat{s}_j$ for all $i \in \mathcal{P}_k$
  for each minibatch of $\mathcal{P}_k$ **do**
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}_{\text{KKT-DPO}}(\theta)$ ▷ Eq. 18
return π_θ

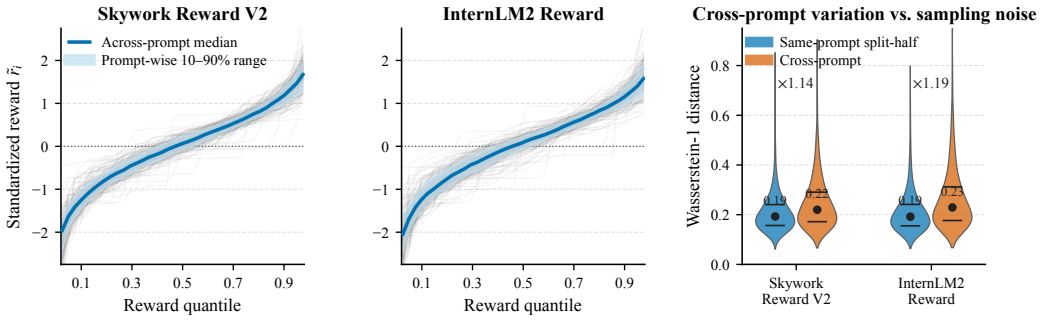
C.4 Algorithm

Algorithm C.1 gives the full training procedure corresponding to the implementation described above and to the objective in Eq. 18.

D Standardized Reward Distributions Across Prompts

Corollary 3.4 gives an exact proportionality when prompts share the same standardized reward distribution. This section examines whether that condition is a reasonable approximation under the reference policy. The analysis is separate from DPO and KKT-DPO training and does not use s_i, $\hat{s}_i$, β_i, or downstream evaluation results.

We select 256 prompts from the test split of `princeton-nlp/llama3-ultrafeedback`. The prompts are ranked by tokenized length under the reference tokenizer and divided into four empirical length quartiles, from each of which we sample 64 prompts without replacement using seed 20270915. For every prompt, we independently sample 64 responses from the frozen Meta-Llama-3-8B-Instruct reference checkpoint, giving 16,384 responses in total. Sampling uses temperature 1.0, top-p 1.0, no top-k truncation, repetition penalty 1.0, and at most 1,024 new tokens. Each response is generated in a separate call with a deterministic prompt–sample-specific seed. Duplicate responses are retained because their frequencies represent probability mass under the empirical reference distribution.



(a) (b) (c)

Figure D.1: Standardized reward distributions across prompts for the Llama-3-8B-Instruct reference policy. (a) and (b) show empirical quantile functions computed from 64 responses for each of 256 prompts after within-prompt standardization. Thin gray curves denote individual prompts, while the blue curve and shaded band denote the across-prompt median and the pointwise tenth–ninetieth percentile range. (c) compares Wasserstein-1 distances between independently standardized 32-response halves of the same prompt and equally sized halves from different prompts over 100 random repetitions. The cross-prompt median is $1.14\times$ and $1.19\times$ the same-prompt baseline under Skywork Reward V2 and InternLM2 Reward, respectively, indicating that cross-prompt variation in standardized reward distributions is modest relative to sampling variation.

Each response is then scored separately by two fixed external reward models, `Skywork-Reward-V2-Llama-3.1-8B`[13] and `internlm2-7b-reward`[14]. For each model, we encode the complete prompt–response conversation using that model's own tokenizer and chat template and record the raw scalar reward-head output. We apply no sigmoid transformation, normalization, discretization, or averaging across reward models. All subsequent calculations are performed separately for the two reward models.

Let $r_{ij}^{(m)}$ be the score assigned by reward model m to response j for prompt i, where $M = 64$. We standardize the scores independently within each prompt and reward model:

$$z_{ij}^{(m)} = \frac{r_{ij}^{(m)} - \bar{r}_i^{(m)}}{s_i^{(m)}}, \qquad \bar{r}_i^{(m)} = \frac{1}{M}\sum_{j=1}^{M} r_{ij}^{(m)}, \qquad s_i^{(m)} = \left[\frac{1}{M}\sum_{j=1}^{M}\left(r_{ij}^{(m)} - \bar{r}_i^{(m)}\right)^2\right]^{1/2}. \tag{54}$$

Figure D.1(a) and Figure D.1(b) plot the empirical quantile function of $\{z_{ij}^{(m)}\}_{j=1}^{64}$ for every prompt. The thin gray curves represent the 256 individual prompts, while the blue curve and shaded band give the across-prompt median and the pointwise tenth–ninetieth percentile range.

To compare cross-prompt variation with finite-sample variation, Figure D.1(c) constructs a same-prompt sampling baseline. In each of 100 repetitions, the 64 scores of every prompt are divided into two disjoint sets of 32, and each half is standardized using its own mean and standard deviation. For two equal-sized standardized samples A and B, we compute the empirical one-dimensional Wasserstein distance

$$\widehat{W}_1(A, B) = \frac{1}{32}\sum_{j=1}^{32}\left|z_{(j)}^{A} - z_{(j)}^{B}\right|, \tag{55}$$

where $z_{(j)}$ denotes the jth order statistic. The same-prompt distribution compares the two halves of one prompt, whereas the cross-prompt distribution compares the first half of one prompt with the second half of a different prompt.

The median same-prompt and cross-prompt distances are 0.193 and 0.220 under Skywork Reward V2, and 0.192 and 0.229 under InternLM2 Reward. Thus, the cross-prompt median exceeds the

[13] https://huggingface.co/Skywork/Skywork-Reward-V2-Llama-3.1-8B
[14] https://huggingface.co/internlm/internlm2-7b-reward

Table E.1: Mathematics and reasoning benchmarks. A down arrow marks a score below the corresponding SFT model.

Model	Method	MATH	TheoremQA	SVAMP	ASDiv	GSM-Plus	BBH
Llama-3-Instruct (8B)	SFT	26.24	14.12	74.4	75.73	48.70	71.77
	DPO	24.70↓	17.25	78.2	83.33	54.66	72.14
	KKT-DPO	**28.38**	**19.25**	**79.0**	**86.08**	**55.05**	**73.61**
Mistral-Instruct-v0.2 (7B)	SFT	10.28	8.88	46.7	59.39	26.13	60.80
	DPO	11.10	11.50	62.4	67.64	32.09	63.40
	KKT-DPO	**11.62**	**11.63**	**63.7**	**69.58**	**32.44**	**65.23**

Table E.2: Coding and instruction-following benchmarks. We report prompt-level strict accuracy for IFEval; full IFEval results are reported in Table E.3. The best scores are in **bold**, and a down arrow marks a score below the corresponding SFT model. Avg is the arithmetic mean over the ten reasoning, mathematics, coding, and instruction-following tasks reported in Tables E.1 and E.2.

Model	Method	HumanEval	LeetCode	MBPP	IFEval	Avg
Llama-3-Instruct (8B)	SFT	53.66	10.00	51.13	61.74	48.75
	DPO	54.27	11.67	51.13	62.66	51.00
	KKT-DPO	**58.54**	**16.11**	**60.90**	**64.14**	**54.11**
Mistral-Instruct-v0.2 (7B)	SFT	25.61	5.56	31.83	40.48	31.57
	DPO	29.88	3.33↓	32.83	41.77	35.59
	KKT-DPO	**31.71**	**7.22**	**34.84**	**48.43**	**37.64**

same-prompt baseline by only $1.14\times$ and $1.19\times$, respectively. The standardized reward distributions are therefore not identical across prompts, but the remaining cross-prompt variation is modest relative to the finite-sample variation induced by 32 responses. The same pattern appears under two independently scaled reward models. These results support treating the standardized-shape factor Φ_i as approximately shared across prompts in this setting, rather than requiring the exact equality in Corollary 3.4.

E Detailed Experimental Results

E.1 Reasoning, mathematics, coding, and instruction-following tasks

Figure 2 in Section 5.2 summarizes the results on reasoning, mathematics, coding, and instruction-following tasks. Table E.1 and Table E.2 report the corresponding scores in full.

Across the results in Table E.1 and Table E.2, KKT-DPO improves over DPO on every reported reasoning, mathematics, coding, and instruction-following task for both model families. The ten-task average increases from 51.00 to 54.11 for Llama-3 and from 35.59 to 37.64 for Mistral.

E.2 Complete IFEval results

Table E.3 reports all four IFEval metrics for both model families and complements the prompt-level strict scores shown in Table E.2.

Consistent with the summary in Table E.2, KKT-DPO improves over DPO on all reported IFEval metrics.

E.3 Additional open-ended statistics

Table E.4 supplements the open-ended results in Table 1 with the average response length and the standard error of the AlpacaEval raw win rate.

Table E.3: IFEval instruction-following results. P and I denote prompt- and instruction-level metrics, respectively.

Model	Method	P-strict	I-strict	P-loose	I-loose
Llama-3	SFT	61.74	71.94	72.64	80.82
	DPO	62.66	72.78	71.35	79.98
	KKT-DPO	**64.14**	**73.14**	**73.38**	**80.82**
Mistral	SFT	40.48	52.16	48.24	60.55
	DPO	41.77	53.72	48.98	61.39
	KKT-DPO	**48.43**	**59.83**	**54.16**	**64.87**

Table E.4: Average response length and standard error of the raw win rate over the 805 AlpacaEval 2 prompts used in Table 1. Average length is measured in characters; win-rate standard errors are measured in percentage points.

	Llama-3-8B-Instruct		Mistral-7B-Instruct-v0.2	
Method	Avg. length	WR SE (pp)	Avg. length	WR SE (pp)
SFT	1995	1.4622	1666	1.2108
DPO	1991	1.6178	1681	1.2762
IPO	2042	1.5391	1788	1.4427
SimPO	1831	1.5737	1647	1.2602
β-DPO	1986	1.5479	2002	1.4687
ε-DPO	1985	1.6094	1677	1.3976
MallowsPO-θ	1958	1.5899	2000	1.4915
KKT-DPO	1919	1.5887	1846	1.4434

The official evaluator computes the raw win rate as the mean of the prediction-level win/loss/tie scores multiplied by 100, and its standard error as the standard error of those scores multiplied by 100. The length-controlled AlpacaEval 2 metric remains the primary comparison because different methods produce responses of different average lengths. Relative to DPO, KKT-DPO is 72 characters shorter for Llama-3 and 165 characters longer for Mistral.

E.4 Realized-divergence measurement protocol

This section gives the measurement protocol underlying Figure 3 in Section 5.4.

Coefficient sweep. Figure 3(c) and 3(d) measure divergence on a fixed set of 500 prompts drawn with seed 42 from the 1,961-prompt test split of UltraFeedback, held out from training and reused for every checkpoint. For each prompt we draw eight responses from the trained policy by ancestral sampling with temperature 1.0, $\mathrm{top_p} = 1.0$, $\mathrm{top_k} = 0$, and at most 2,048 new tokens, giving 4,000 responses per checkpoint. The reference model is Llama-3-8B-Instruct, and both models score the same generated responses. Log probabilities are accumulated over response tokens only, including EOS and excluding prompt tokens, using natural logarithms. The per-prompt divergence is $K_i = \frac{1}{8}\sum_{m=1}^{8}[\log \pi_\theta(y_{im} \mid x_i) - \log \pi_{\mathrm{ref}}(y_{im} \mid x_i)]$, a sequence-level quantity in nats, and the reported value is the mean of K_i over the 500 prompts without clipping. Model computation uses BF16, and logits are converted to FP32 before the log-probability evaluation. Every response terminates at EOS and none reaches the token limit, so the estimate is free of truncation. We repeat the measurement for DPO at $\beta \in \{0.005, 0.01, 0.05\}$ and for KKT-DPO at $\beta_{\mathrm{anchor}} \in \{0.005, 0.01, 0.05\}$.

Divergence by estimated dispersion. Figure 3(b) uses training prompts rather than the held-out set, because the quintiles are defined by the $\hat{s}_i$ values recorded during the KKT-DPO run. That run covers 59,776 unique prompts of the `llama3-ultrafeedback` train split over 467 steps, which form five pools under the refresh schedule of Section 4.3. Within each pool we sort prompts by the logged $\hat{s}_i$, break ties by row index, split them into quintiles, and merge matching quintiles across pools. We sort by $\hat{s}_i$ rather than by β_i, since the training-time coefficient also reflects warmup and a per-step median. We then draw 100 prompts per quintile with seed 42 without replacement. DPO

Table F.1: Sensitivity to the main KKT-DPO hyperparameters on Llama-3-8B-Instruct. Unless varied, the defaults are $\beta_{\text{anchor}} = 0.01$, $\alpha = 2/3$, $N = 100$, and $n = 5$.

Hyperparameter	Value	AE2 LC	AE2 WR
β_{anchor}	0.005	42.44	41.07
	0.01	**44.65**	**43.91**
	0.05	38.86	37.81
α	0	40.88	40.01
	$1/3$	42.30	41.91
	$2/3$	**44.65**	**43.91**
N	1	40.39	39.87
	50	41.92	40.06
	100	**44.65**	**43.91**
n	1	41.69	40.88
	3	42.54	41.64
	5	**44.65**	**43.91**

and KKT-DPO are evaluated on the same 500 prompts under the same quintile assignment, both at coefficient 0.01 and at the same training step. Responses and log probabilities follow the protocol above. Each curve is normalized by its own mean over these prompts, so the panel compares the shape of the allocation rather than its level. The two checkpoints are matched by training step and not by divergence budget.

Coefficient trace. Figure 3(a) is taken directly from the training log of the same KKT-DPO run and contains one β_i value per covered prompt. The plotted trace is an evenly spaced subset together with the observed extrema, chosen for legibility at this figure size.

F Hyperparameter Sensitivity

This section reports sensitivity analyses for the main KKT-DPO hyperparameters on Llama-3-8B-Instruct. Unless varied, the defaults are $\beta_{\text{anchor}} = 0.01$, $\alpha = 2/3$, $N = 100$, and $n = 5$. Table F.1 summarizes all results.

F.1 Anchor and shrinkage

Table F.1 shows that the preferred anchor scale is consistent with the coefficient selected for vanilla DPO. Increasing β_{anchor} from 0.005 to 0.01 improves both metrics, whereas 0.05 lowers AE2 LC by 5.79 points relative to the default.

The same table also shows that substantial shrinkage is beneficial when only five reference responses are available per prompt. Without shrinkage, sampling noise enters the coefficients directly and AE2 LC is 3.77 points below the default. The intermediate value $\alpha = 1/3$ also underperforms the default.

F.2 Snapshot refresh and reference samples

Table F.1 further shows that refreshing the lagged snapshot too frequently is less effective in this setting. Refreshing every step gives an AE2 LC score 4.26 points below the default, and among the tested values $N = 100$ performs best.

Finally, performance improves across the tested numbers of cached reference responses per prompt. The $n = 1$ configuration makes the within-prompt standard deviation zero and therefore recovers vanilla DPO, matching the corresponding ablation in Table 2. Among the tested values, $n = 5$ performs best. Larger values were not evaluated and would increase both offline generation cost and lagged-policy scoring cost.

G COMPUTATIONAL COST

All timing measurements use 8 AMD Instinct MI308X GPUs with 192 GB of memory per GPU.

KKT-DPO does not add a second backward pass or otherwise change the DPO gradient computation. Its measured training time includes periodic no-gradient scoring by the lagged policy snapshot. Vanilla DPO training takes 4.959 wall-clock hours (39.672 GPU-hours), whereas KKT-DPO training takes 5.882 wall-clock hours (47.056 GPU-hours). Relative to vanilla DPO, the measured training increase is therefore 0.923 wall-clock hours.

Reference-policy sampling is a separate offline preprocessing step. It takes approximately 20 wall-clock hours, or 160 GPU-hours, and is performed once for a given reference model and set of training prompts. The coefficient calculation itself is negligible relative to generation and model scoring.

The open-ended training sets already release five on-policy responses per prompt (Appendix C), and those responses are drawn from the same reference model that KKT-DPO uses. Reusing them in Eq. 15 would therefore meet the sampling requirement of Eq. 14 and would remove the 160 GPU-hours of offline generation entirely. We nevertheless draw fresh responses from π_{ref}, for three reasons. First, the released preference pair (y_w, y_l) is selected from those same five responses, so reusing them would make the readout share responses with the loss it scales, whereas independent samples keep the coefficient estimate separate from the pair it is applied to. Second, reuse caps the number of cached responses at five and fixes their decoding configuration, whereas sampling leaves both n and the decoding configuration under our control and is what allows the sensitivity to n reported in Table 2 and Appendix F. Third, a preference corpus need not supply on-policy responses at all: `UltraInteract_pair` in the reasoning, mathematics, coding, and instruction-following setting does not, so sampling from the reference model keeps a single pipeline across both settings. We report this cost in full so that the two options can be compared. Because Eq. 15 constrains only the policy from which the cached responses are drawn, a practitioner whose preference dataset already supplies responses from the reference model in use can skip this step without altering the method.