FUTURE-ALIGNED REINFORCEMENT LEARNING FOR ASYNCHRONOUS LANGUAGE MODEL POST-TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Asynchronous reinforcement learning (RL) accelerates large language model (LLM) post-training by overlapping rollout generation with policy optimization, at the cost of *staleness*: a rollout is generated by a behavior policy that is *one optimizer update behind* the target policy by the time it is consumed. Most existing methods address this staleness *reactively*, for example through importance-sampling and clipping schemes or distribution-matched replay frameworks, repairing stale data after it has been produced. We introduce **Future-Aligned Reinforcement Learning (FARL)**, a *proactive* method that, at rollout-dispatch time, forecasts the consumption-time policy (the learner one update ahead) and generates the entire trajectory from this single forecasted, frozen worker policy. The forecast extrapolates the learner's next parameter displacement from the most recent updates using a consistency-constrained multi-lag predictor. We show that the consistency constraint cancels the first-order constant-drift component of policy lag: under fixed-step smooth learner dynamics this yields an *asymptotic* $\mathcal{O}(\eta^2)$ upper bound on the forecast residual against $\mathcal{O}(\eta)$ for stale generation, where η is the learner's step size, and correspondingly tightens the local bound on the variance of the exact importance weights $\pi_{\text{target}}/\pi_{\text{behavior}}$ from $\mathcal{O}(\eta^2)$ to $\mathcal{O}(\eta^4)$. For finite-step stochastic training we give a separate coefficient-aware data-dependent bound and measure the realized prequential forecast error directly. Because FARL acts on the behavior policy *before* generation rather than repairing data afterward, it is a drop-in *plug-in module* orthogonal to existing reactive correctors, and across a thorough set of experiments it improves the average accuracy of every one of them. Code is available at `https://anonymous.4open.science/r/farl-anonymous-0442`.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) has become a central tool for eliciting long chain-of-thought reasoning from large language models (LLMs), with group-relative policy optimization (GRPO) as the dominant optimizer (Shao et al., 2024). In the *synchronous* regime, every optimization step must wait for a fresh batch of on-policy rollouts. Because a single reasoning trajectory can span thousands of decoded tokens, the trainer's expensive accelerators sit idle during generation, and this generation–optimization stall dominates wall-clock cost.

Asynchronous RL removes the stall by separating rollout generation from policy optimization: the generator keeps producing rollouts while the trainer keeps updating the policy, instead of each waiting for the other. This overlap keeps the accelerators busy and improves training throughput (Noukhovitch et al., 2025; Fu et al., 2026; Lu et al., 2025; Uehara et al., 2022). To keep the behavior policy as close as possible to the policy being optimized, existing asynchronous methods train each update on samples generated by the *most recent* behavior policy, that is, the rollouts a worker produces are consumed *one* optimizer update later (Hilton et al., 2022; Bai et al., 2026). This *one-step-off* regime, in which the learner advances exactly one update between dispatch and consumption, is the standard asynchronous setting and the setting we study throughout this paper (we write the delay as d=1; the analysis extends to general d, but all main experiments use d=1; Appendix O sweeps $d \in \{1, 2, 3, 4\}$). Concretely, a rollout dispatched when the learner is at parameters θ_s is consumed one update later at θ_{s+1}, so the behavior policy π_{θ_s} that produced the data no longer matches the target policy $\pi_{\theta_{s+1}}$ being updated. This induces *staleness*: a fundamental distributional

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.

mismatch between the off-policy data and the current policy that injects high-variance off-policy gradients and degrades training when left uncontrolled (Noukhovitch et al., 2025; Bai et al., 2026; Espeholt et al., 2018).

A growing body of work mitigates staleness *reactively*: it generates stale data and then repairs it before or during the gradient step. These correctors fall into two families. *Importance-sampling and clipping* methods reweight or bound the off-policy ratio at the update. CISPO (Chen et al., 2025) clips the importance-sampling ratio and applies a stop-gradient to the clipped weight, so every token continues to contribute a gradient rather than being zeroed out as under standard PPO/GRPO clipping; decoupled PPO (Hilton et al., 2022) introduces an auxiliary proximal policy and splits the ratio into short- and long-term factors to curb the variance induced by policy drift; tapered off-policy REINFORCE (Roux et al., 2025) smoothly downweights large ratios; and truncated importance weighting (Espeholt et al., 2018) caps them outright. *Distribution-matched replay* methods instead curate *which* stale samples to train on. The closest work to ours, **D-ARL** (Bai et al., 2026), maintains a replay buffer over the most recent K behavior policies and trains on a distribution-matched, minimum-variance selection of the buffered samples. Despite their differences, all of these methods share one structure: *generate stale data, then repair it.*

In this paper, we ask a different question: *can we avoid generating stale data in the first place?* Our answer is **Future-Aligned Reinforcement Learning (FARL)**, a *proactive* alternative that attacks staleness at its source (Figure 1). When a worker begins a rollout at θ_s that will be consumed one learner update later at θ_{s+1}, FARL forecasts the consumption-time displacement $\widehat{S}_{s,1} \approx \theta_{s+1} - \theta_s$ and generates the entire trajectory from the single, frozen, forecasted worker policy $\pi_{\theta_s + \widehat{S}_{s,1}}$, converting "generate stale data and repair it afterward" into "predict where the learner will be and generate data for that destination." The forecast comes from a consistency-constrained multi-lag predictor over the m most recent updates, fit by a delayed, leakage-free online ridge calibration, so a rollout never depends on its own future checkpoint, reward, or gradient. Theoretically, the consistency constraint cancels the first-order constant-drift component of policy lag: under fixed-step smooth state-based learner dynamics, stale generation admits an $\mathcal{O}(\eta)$ upper bound on the behavior–target parameter mismatch and every consistency-constrained forecast an $\mathcal{O}(\eta^2)$ bound, where η is the learner's step size; under local order-2 Rényi regularity these induce $\mathcal{O}(\eta^4)$ versus $\mathcal{O}(\eta^2)$ upper bounds on the variance of the exact importance weights $\pi_{\text{target}}/\pi_{\text{behavior}}$, a quantity closely related to the stability of exact off-policy correction. These are *local, asymptotic* bounds and do not imply finite-step variance dominance; for that regime the same theorem supplies a sharper realized-update bound depending on the observed learner trajectory, and we measure the realized forecast error directly (Appendix P). Crucially, only the rollout-worker copy is forecast—the learner parameters, optimizer, reward, and objective are untouched—so FARL is a source-side *plug-in module* that is orthogonal to every reactive corrector above and can be stacked on any of them. As we show, it improves the average accuracy of each one (Figure 2).

Our contributions are as follows.

• **A proactive, leakage-free method (§3).** FARL replaces the stale behavior policy with a forecast of the consumption-time policy one update ahead, built from a consistency-constrained multi-lag predictor over recent learner updates. Its coefficients are fit online by a delayed, forgetting-factor ridge regression that uses only origins whose targets have already matured, so a rollout never depends on its own future checkpoint, reward, or gradient. The forecast is the only change to the pipeline: the learner parameters, optimizer, objective, and reward are untouched.

• **Theory (§4).** We prove that the forecast error is governed by the *variation* of consecutive learner updates rather than their total magnitude, in two forms: a finite-step bound depending only on the realized update differences along the observed trajectory, and, under fixed-step smooth dynamics, an asymptotic $\mathcal{O}(\eta^2)$ upper bound for every consistency-constrained forecast against $\mathcal{O}(\eta)$ for stale generation—which under local order-2 Rényi regularity carries over to the second moment and variance of exact trajectory importance weights. We further show that an unconstrained forecast-strength scalar would reintroduce first-order error, that the calibrated predictor cannot fit the matured history worse than constant velocity, and that a one-version lag error adds at most a first-order term.

• **A plug-in module validated at scale (§5).** Because FARL acts on the behavior policy before generation, it is orthogonal to the reactive mechanisms of existing async-RL methods and can be stacked on any of them. We add FARL to a representative method from each family—importance-sampling

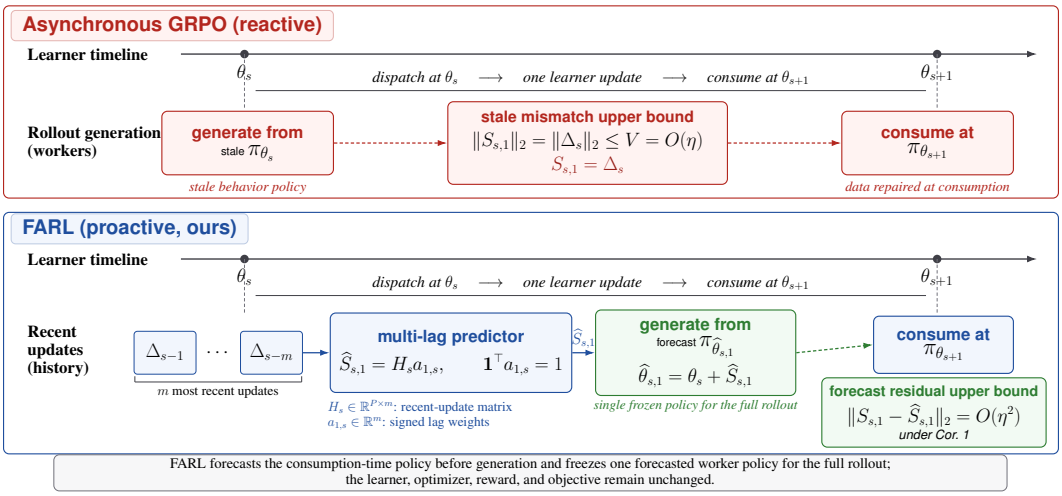

Figure 1: *Top:* asynchronous GRPO generates from the stale policy and repairs the mismatch afterwards. *Bottom:* FARL extrapolates recent learner updates to predict the policy that will consume the rollout, and generates from that single frozen forecast. Shown at the single-step lag (d=1).

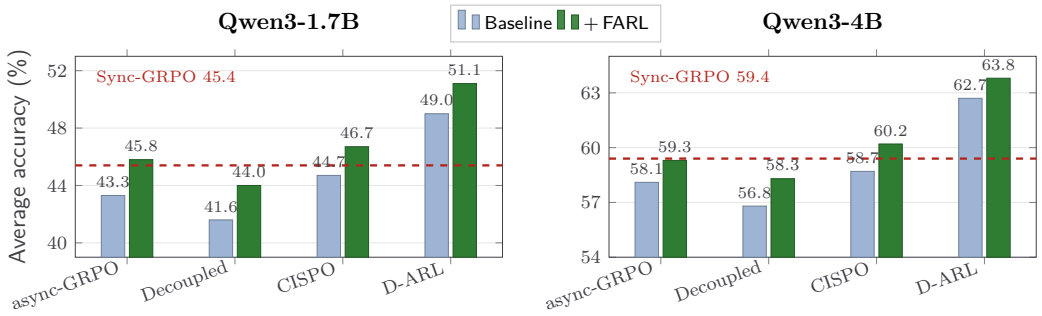

Figure 2: **FARL is a plug-in that raises the average accuracy of every asynchronous baseline**, on both model scales. Bars show the mean accuracy over the nine benchmarks (the **AVG** column of Table 1) for each base method (light blue) and the same method with FARL stacked on top (green). The dashed red line marks the no-lag synchronous-GRPO reference.

clipping (CISPO (Chen et al., 2025)), gradient decoupling (Decoupled (Hilton et al., 2022)), and distribution-matched replay (D-ARL (Bai et al., 2026))—and to plain asynchronous GRPO, evaluating on nine reasoning benchmarks spanning mathematics and code at two model scales over three seeds. FARL improves the nine-benchmark average of every method at both scales and 69 of the 72 individual method–benchmark results; the three exceptions are the most saturated math sets for the strongest 4B baseline, where it is essentially tied (-0.2 each).

2 Background and Related Work

We focus on the work most directly related to FARL. Broader background on RLVR, actor–learner asynchrony, and asynchronous LLM-RL *systems* is deferred to Appendix D.

Reactive correction of stale LLM rollouts. Most algorithmic approaches accept the stale behavior distribution and modify the learner update afterward. One line adjusts the objective: Decoupled PPO (Hilton et al., 2022) separates the behavior-policy correction from the proximal trust-region anchor, off-policy PPO (Meng et al., 2023) develops a PPO objective for replayed experience, and A-3PO (Li et al., 2026) approximates the decoupled proximal policy by interpolation to avoid an extra forward pass. A second controls the importance-sampling ratio itself: CISPO (Chen et al., 2025) clips the weight to a trust-region interval (asymmetric in general) and applies a stop-gradient to it, keeping a gradient on every token; tapered off-policy REINFORCE (Roux et al., 2025) trades bias for variance; and truncated importance sampling (Espeholt et al., 2018) caps the ratio outright,

building on classical off-policy estimators and their variance analysis (Hanna et al., 2021; Uehara et al., 2022). GEPO (Zhang et al., 2026) controls importance-weight variance under high-latency training, and recent scaling-law analysis relates stale-surrogate bias to the product of update size and rollout delay (Song et al., 2026). All of these are reactive: they alter how an already-generated sample contributes to the learner. FARL is complementary because it changes the source distribution before the sample exists.

Replay, sample selection, and multi-behavior learning. D-ARL (Bai et al., 2026), our primary reactive baseline, stores trajectories from the most recent K behavior policies and trains on a distribution-matched, minimum-variance selection of them, observing that temporally old samples can remain distributionally useful, so age alone is an imperfect proxy for suitability. Other replay mechanisms instead bound age or prioritize informative trajectories (Fu et al., 2026; Lu et al., 2025; Yoo et al., 2026). FARL is orthogonal to all of these: it forecasts each producer policy before generation, after which replay selection and multi-behavior aggregation apply unchanged with the forecast policy as the behavior source.

Predictive compensation and lookahead under delay. The closest conceptual precedents arise in delayed optimization rather than LLM rollout generation: delay-compensated ASGD corrects stale gradients by a local Taylor approximation (Zheng et al., 2017), momentum lookahead compensates stale updates in asynchronous low-communication language-model training (Ajanthan et al., 2025), and asynchronous actor–critic architectures decouple acting from learning and must cope with the resulting policy lag (Mnih et al., 2016; Xiao et al., 2022). These methods compensate a gradient or model update that has already been computed. FARL predicts a different object at an earlier stage: before an expensive language trajectory exists, it forecasts the learner displacement at the expected consumption version and uses that forecast as the frozen behavior policy for generation, with a consistency constraint that cancels the first-order constant-drift component of policy lag.

3 Method: Future-Aligned Reinforcement Learning

3.1 Problem setup: policy staleness in asynchronous RL

Let π_{θ_t} denote the learner policy after optimization step t, where $\theta_t \in \mathbb{R}^P$, and define the realized learner update

$$\Delta_t \coloneqq \theta_{t+1} - \theta_t. \tag{1}$$

Appendix A (Table 2) collects all symbols used below for reference. Throughout, π_θ denotes the *actual normalized rollout distribution* induced by the model and its decoding configuration: any temperature scaling is included in π_θ. A rollout dispatched when the learner is at θ_s may be consumed after d additional learner updates. The consumption-time policy is therefore $\pi_{\theta_{s+d}}$, with

$$\theta_{s+d} = \theta_s + S_{s,d}, \qquad S_{s,d} \coloneqq \sum_{r=0}^{d-1} \Delta_{s+r}. \tag{2}$$

Conventional asynchronous GRPO generates the rollout from the dispatch-time policy π_{θ_s}. Consequently we define parameter staleness as $\|S_{s,d}\|_2$, which accumulates the full learner movement during rollout generation and queueing. In our asynchronous pipeline, and in *every* experiment in this paper, the consumption lag is a single optimizer step (d=1), so the consumption-time policy is $\pi_{\theta_{s+1}}$ and the staleness reduces to $S_{s,1} = \Delta_s$. We nonetheless carry a general d through the derivations below, since the construction and its analysis are identical for any fixed lag.

FARL instead constructs, using only information available at dispatch time, a prediction $\widehat{S}_{s,d}$ of the future learner displacement $S_{s,d}$ and generates from

$$q_{s,d} \coloneqq \pi_{\widehat{\theta}_{s,d}}, \qquad \widehat{\theta}_{s,d} \coloneqq \theta_s + \widehat{S}_{s,d}. \tag{3}$$

Only the rollout-worker copy is forecast. The learner parameters, optimizer, reward, and policy-optimization objective are unchanged. The forecasted worker policy is frozen for the entire trajectory, and its token-level behavior log probabilities are stored for learner-side off-policy correction.

We maintain a separate predictor for each consumption lag $d \in \{1, \ldots, D_{\max}\}$. In a fixed-lag pipeline, d is known; our experiments use the one-step pipeline throughout, so only the $d{=}1$ predictor is ever exercised. In a variable-lag system, the scheduler predicts a version lag $\widehat{d}_s$ from queue state and measured actor/learner throughput. Section 4 quantifies the additional first-order error caused by lag misspecification.

3.2 Consistency-constrained multi-lag prediction

FARL predicts future learner movement from the m most recent completed learner updates. Define

$$H_s \coloneqq [\Delta_{s-1} \quad \Delta_{s-2} \quad \cdots \quad \Delta_{s-m}] \in \mathbb{R}^{P \times m}. \tag{4}$$

For horizon d, the raw forecast is

$$\widehat{S}_{s,d} = H_s a_{d,s} = \sum_{j=1}^{m} a_{d,s,j} \Delta_{s-j}, \qquad a_{d,s} \in \mathbb{R}^m. \tag{5}$$

The coefficients are signed: positive weights represent persistence, whereas negative weights permit curvature and oscillation corrections. Crucially, we impose the first-order consistency constraint

$$\mathbf{1}^\top a_{d,s} = d. \tag{6}$$

If the observed updates are constant, $\Delta_{s-j} = \Delta$ for all $j = 1, \ldots, m$, then the consistency constraint yields $H_s a_{d,s} = d\Delta$. If the learner continues with the same update over the next d steps, namely $\Delta_{s+r} = \Delta$ for $r = 0, \ldots, d-1$, then $S_{s,d} = d\Delta$ and the forecast is exact. Thus $\mathbf{1}^\top a_{d,s} = d$ makes the predictor exact under constant-velocity learner dynamics and cancels the corresponding first-order component under slowly varying dynamics.

To prevent unstable extrapolation, coefficients are restricted to

$$\mathcal{A}_d \coloneqq \left\{ a \in \mathbb{R}^m : \mathbf{1}^\top a = d,\ \|a\|_2 \le B_d \right\}, \qquad B_d \ge d, \tag{7}$$

so that the constant-velocity predictor

$$a_d^{\mathrm{cv}} \coloneqq d e_1, \qquad H_s a_d^{\mathrm{cv}} = d\Delta_{s-1}, \tag{8}$$

is always feasible. Unlike post-hoc projection of the predicted displacement, the coefficient constraint preserves Equation 6 and therefore preserves the second-order cancellation proved below.

Note that the adaptive vector $a_{d,s}$ controls both the direction and magnitude of the forecast.

3.3 Leakage-free delayed online calibration

The target $S_{i,d}$ for a forecast made at origin i becomes available only when the learner reaches step $i + d$. At learner step t, define the matured origins

$$\mathcal{I}_{d,t} \coloneqq \{ i :\ i \ge m,\ i + d \le t \}. \tag{9}$$

Forgetting factor $\kappa \in (0, 1]$ allows the predictor to track nonstationary optimizer dynamics and is unrelated to any reward discount factor. FARL computes

$$a_{d,t} \in \arg\min_{a \in \mathcal{A}_d} \left\{ \sum_{i \in \mathcal{I}_{d,t}} \kappa^{\,t-(i+d)} \|S_{i,d} - H_i a\|_2^2 + \lambda \|a - a_d^{\mathrm{cv}}\|_2^2 \right\}, \tag{10}$$

where $\lambda > 0$ shrinks the learned predictor toward constant velocity when few matured targets are available. Since $a_d^{\mathrm{cv}} = d e_1 \in \mathcal{A}_d$, the feasible set is nonempty; moreover $\mathcal{A}_d$ is compact and convex, and $\lambda > 0$ makes the objective strongly convex. Hence the calibrated coefficient vector $a_{d,t}$ exists and is unique. All examples used in Equation 10 correspond to forecasts whose outcomes are already known; hence the current rollout never uses its own future checkpoint, reward, or gradient.

The solution is available in closed form. Define the ridge-regularized Gram matrix and target vector

$$M_{d,t} \coloneqq \sum_{i \in \mathcal{I}_{d,t}} \kappa^{\,t-(i+d)} H_i^\top H_i + \lambda I, \qquad c_{d,t} \coloneqq \sum_{i \in \mathcal{I}_{d,t}} \kappa^{\,t-(i+d)} H_i^\top S_{i,d} + \lambda a_d^{\mathrm{cv}}, \tag{11}$$

which are the only quantities the regression stores beyond the update history—an $m \times m$ matrix and an m-vector per delay bucket—and which admit an exact streaming recursion (Appendix F). Since $\lambda > 0$ we have $M_{d,t} \succ 0$, and ignoring only the norm constraint in Equation 7, the solution is the $M_{d,t}$-metric projection of the unconstrained ridge estimate onto the consistency hyperplane $\mathbf{1}^\top a = d$:

$$a_{d,t}^{(0)} = M_{d,t}^{-1} c_{d,t}, \qquad \widetilde{a}_{d,t} = a_{d,t}^{(0)} - M_{d,t}^{-1}\mathbf{1}\,\frac{\mathbf{1}^\top a_{d,t}^{(0)} - d}{\mathbf{1}^\top M_{d,t}^{-1}\mathbf{1}}. \tag{12}$$

If $\|\widetilde{a}_{d,t}\|_2 \le B_d$ then $a_{d,t} = \widetilde{a}_{d,t}$; otherwise Equation 10 is a convex quadratic program in only m variables and is solved directly. Before any target has matured, and during the warm-up steps $s < m$ where fewer than m updates exist, FARL falls back to the constant-velocity prior a_d^{cv} rather than zero-padding artificial update vectors (Appendix F).

3.4 Version-aware dispatch and behavior-policy logging

At dispatch step s, the scheduler supplies the predicted consumption lag $\widehat{d}_s$. The worker constructs

$$\widehat{\theta}_{s,\widehat{d}_s} = \theta_s + H_s a_{\widehat{d}_s,s} \tag{13}$$

and generates the full response group under the single fixed policy $\pi_{\widehat{\theta}_{s,\widehat{d}_s}}$. Each trajectory stores (i) the dispatch policy version s, (ii) the predicted lag $\widehat{d}_s$, (iii) the exact behavior-policy token log probabilities, and (iv) the eventual consumption version. These records permit direct measurement of prediction error as well as exact evaluation of behavior-to-target likelihood ratios.

When the rollout is consumed at learner step t, FARL uses the same reward, group-relative advantage, clipping rule, and optimizer as the asynchronous GRPO baseline. For a generated token $y_{i,k}$, the ratio is computed against the actual forecasted behavior policy:

$$\rho_{i,k}(\theta_t) = \frac{\pi_{\theta_t}(y_{i,k} \mid x, y_{i,<k})}{\pi_{\widehat{\theta}_{s,\widehat{d}_s}}(y_{i,k} \mid x, y_{i,<k})}. \tag{14}$$

Thus FARL changes only the data-generation distribution. It does not alter the learner-side asynchronous-GRPO surrogate. In particular, the practical GRPO surrogate clips these token-level, whereas the change-of-measure analysis of Theorem 3 concerns the exact, untruncated trajectory ratio $W(y) = \prod_k \rho_{i,k}$; we do not claim that the clipped GRPO surrogate is unbiased. The full procedure at each learner step t is summarized in Algorithm 1 (Appendix F); the behavior policy is the only object it changes, and the learner update (line 13) is the unmodified asynchronous-GRPO or D-ARL objective.

4 Theoretical Analysis

For readability, all proofs in this section are deferred to Appendix C.

4.1 Consistency and variation-controlled forecast error

Assumption 1 (Bounded learner updates and update variation)**.** *There exist constants $V, A \ge 0$ such that, for every index appearing below,*

$$\|\Delta_t\|_2 \le V, \qquad \|\Delta_{t+1} - \Delta_t\|_2 \le A. \tag{15}$$

Theorem 1 (Consistency-constrained multi-lag forecast)**.** *Fix a dispatch version s, a horizon $d \ge 1$, and a coefficient vector $a = (a_1, \dots, a_m)^\top \in \mathcal{A}_d$, where $\mathbf{1}^\top a = d$ and $\|a\|_2 \le B_d$, with H_s and $S_{s,d}$ as in Equations 2 and 4. Then the consistency constraint $\mathbf{1}^\top a = d$ yields the exact decomposition*

$$S_{s,d} - H_s a = \sum_{r=0}^{d-1} \big(\Delta_{s+r} - \Delta_s\big) - \sum_{j=1}^{m} a_j \big(\Delta_{s-j} - \Delta_s\big), \tag{16}$$

in which every term is a difference *of learner updates rather than an update itself. Consequently the forecast residual obeys the finite-step, trajectory-specific bound*

$$\|S_{s,d} - H_s a\|_2 \le \sum_{r=0}^{d-1} \|\Delta_{s+r} - \Delta_s\|_2 + \sum_{j=1}^{m} |a_j| \, \|\Delta_{s-j} - \Delta_s\|_2. \tag{17}$$

If in addition Assumption 1 holds, then $\|\Delta_{s+r} - \Delta_s\|_2 \le rA$ *and* $\|\Delta_{s-j} - \Delta_s\|_2 \le jA$*, so with* $C_{d,m} \coloneqq \frac{d(d-1)}{2} + B_d\sqrt{m(m+1)(2m+1)/6}$*,*

$$\|S_{s,d} - H_s a\|_2 \le A \left[\frac{d(d-1)}{2} + \sum_{j=1}^{m} j|a_j| \right] \le A\, C_{d,m}, \tag{18}$$

whereas stale generation from θ_s *satisfies*

$$\|S_{s,d}\|_2 \le dV. \tag{19}$$

Corollary 1 (Fixed-step smooth state-based updates)**.** *Suppose* $\Delta_t = \eta g(\theta_t)$*, where* $\eta > 0$ *is fixed,* g *is L-Lipschitz, and* $\|g(\theta)\|_2 \le G$ *throughout the relevant region. Then* $V \le \eta G$ *and* $A \le \eta^2 LG$*, so for fixed* d*,* m*, and* B_d*,*

$$\|S_{s,d}\|_2 \le d\eta G = O(\eta), \qquad \|S_{s,d} - H_s a\|_2 \le \eta^2 LG\, C_{d,m} = O(\eta^2) \quad \textit{for every } a \in \mathcal{A}_d. \tag{20}$$

These are upper bounds: the result does not assert that the realized stale mismatch is $\Theta(\eta)$ *or that the realized FARL mismatch is always smaller at a finite step. It is a* sufficient asymptotic regime *exhibiting the cancellation, not a condition we claim holds exactly for stochastic GRPO: under* $\Delta_t = \eta\,(g(\theta_t) + \xi_t)$ *with unpredictable innovations* ξ_t *the realized residual need not be* $O(\eta^2)$*, and the memoryless model is likewise illustrative rather than verbatim true for stateful optimizers such as Adam (Appendix B).*

What $C_{d,m}$ does and does not certify. Equation 18 is a worst-case corollary of the realized bound Equation 17, and a loose one: at our operating point it improves on the stale bound dV only when $A/V \lesssim 0.06$, against realized ratios of 0.72–0.91. We therefore do *not* use $A\,C_{d,m}$ to explain the finite-step experiments; its role is structural, exhibiting the order cancellation bought by the consistency constraint. The finite-step behavior is instead characterized by the realized bound Equation 17, which at d=1 sharpens into a data-dependent certificate, and is measured directly on the training trajectory (Appendix P).

Further results (Appendix B). Three results, proved in Appendix C, sharpen this bound: the consistency constraint is what buys the second order, since a free $O(1)$ forecast-strength scalar can reintroduce a first-order error (Proposition 1); the online ridge calibration cannot fit the matured history worse than the constant-velocity prior it shrinks toward (Proposition 2); and a mispredicted consumption lag adds only a first-order $O(|d - \widehat{d}|\eta)$ term (Proposition 3).

The parameter-space error also controls the *exact* trajectory importance weights that asynchronous RL reweights by: under local order-2 Rényi regularity the second moment and variance of $W = \mathrm{d}p/\mathrm{d}q$ are bounded by $\exp\big(C_\pi \|S_{s,d} - H_s a\|_2^2\big)$ (Theorem 3). Since Euclidean parameter distance is not itself a policy metric, we also expand in the trajectory Fisher information: the KL divergence, importance-weight variance, effective sample size, and clipping rate between behavior and target are all governed by the Fisher-weighted error $e^\top F_x e$ (Theorem 2; a token-level version matching the implemented clipped GRPO surrogate is Theorem 4). Both routes give $O(\eta^4)$ for FARL against $O(\eta^2)$ for stale generation (Corollaries 3 and 2), under the same local, asymptotic assumptions as Corollary 1; they should not be read as finite-step variance dominance. Appendix K (Table 8) reports all four measured metrics, where the parameter-space advantage carries over to function space at both scales, and explains why the policy-space reduction in fact *exceeds* the parameter-space one—the same Euclidean-versus-Fisher distinction rather than a discrepancy.

5 Experiments

5.1 Training setup and hyperparameters

We train two base models, **Qwen3-1.7B** and **Qwen3-4B**, under a one-step-off asynchronous schedule (d=1), and set m=4 in FARL. We use two training corpora, both from Guru-RL (Cheng et al.,

2026): a *mathematics* corpus (the 6,200-problem math subset) and a *code* corpus (the 18,085-problem code subset). Appendix E gives the full training setup, the remaining FARL hyperparameters, and the shared trust-region configuration, with the complete list in Table 3.

Baselines. All baselines share the same backbone, rollout budget, reward, KL penalty, and optimizer, and differ only in how they handle staleness. **Sync-GRPO** is synchronous GRPO with no lag, a no-staleness reference point; **Async-GRPO** is the one-step-off variant trained directly on stale rollouts with no *staleness-specific* correction. It is not an uncorrected objective: every arm, including the zero-lag Sync-GRPO reference, uses the same clipped GRPO surrogate with truncated importance sampling at threshold 2.0 (Appendix E), so that surrogate is a controlled constant rather than a treatment. The three reactive correctors are **CISPO** (Chen et al., 2025) (stop-gradient importance-ratio clipping), **Decoupled** (Hilton et al., 2022) (an auxiliary proximal policy with a short/long-term ratio split), and **D-ARL** (Bai et al., 2026) (the strongest baseline, using distribution-matched replay over the most recent $K=4$ behavior policies); see Section 2 for details. Because FARL only replaces the stale behavior policy with a forecast of the consumption-time policy, it is orthogonal to all of these; we report it standalone (Async-GRPO vs. FARL) and stacked on each baseline (Variant vs. FARL+Variant), with per-method integration in Appendix G.

Benchmarks and evaluation protocol. We evaluate on nine reasoning benchmarks: grade-school math GSM8K (Cobbe et al., 2021), the MATH-500 subset (Lightman et al., 2024) of the MATH dataset (Hendrycks et al., 2021) and its full 5,000-problem MATH test set (LightEval), the competition sets AIME24, AIME25, AMC23, and AMC24, and the code benchmarks HumanEval (Chen et al., 2021) and LiveCodeBench (Jain et al., 2025); test-set sizes are listed in Table 4. All evaluation uses rollout temperature 1.0. We report avg@5 accuracy (5 samples per problem) for GSM8K, MATH-500, LightEval, HumanEval, and LiveCodeBench, and avg@16 for the small competition sets, whose much smaller test sets make low-sample estimates high-variance. Training data and a contamination check for the code benchmarks are described in Appendix E.

5.2 Consolidated results across all benchmarks

Table 1 reports accuracy (%) averaged over three seeds, and Figure 5 shows that almost all gains are statistically significant under a two-sided 95% Welch confidence interval. The improvement also holds *throughout* training, not just at the final checkpoint: across all four baselines and all nine benchmarks, the +FARL curve sits at or above its baseline for essentially the entire run (Appendix I, Figure 4).

FARL improves every asynchronous baseline (Qwen3-1.7B). On the smaller model (Table 1, top block; averages in Figure 2), the forecast improves *every* method it is stacked on, on *every* benchmark, without a single regression. Averaged over the nine benchmarks it lifts Async-GRPO by +2.5 (43.3→45.8), Decoupled by +2.4, CISPO by +2.0, and D-ARL by +2.1 (49.0→51.1), with the largest individual gains on the least-saturated sets (+5.3 on HumanEval and +3.7 on LiveCodeBench over D-ARL). The best configuration, **FARL+D-ARL**, attains the top score in every column; at 51.1 average it exceeds the synchronous no-lag reference (Sync-GRPO, 45.4). The measured cost of one-step staleness is 2.1 points (45.4 → 43.3), and standalone FARL recovers +2.5, so the forecast more than closes the gap to synchronous training. These gains are large relative to seed-to-seed variation, which stays at or below ±0.5 on the nine-benchmark average (Appendix J).

Scaling to Qwen3-4B. The same pattern holds at 4B (Table 1, bottom block), with FARL improving all four baselines by +1.1 to +1.5, but the margins compress: the 4B model has a *smaller* realized behavior–target KL at $d=1$ (Table 8) and its math benchmarks are near saturation, leaving the forecast less headroom. **FARL+D-ARL** is essentially tied with D-ARL on the three most saturated math sets (MATH-500, AIME24, AIME25: −0.2 each) and gains on the remaining seven columns, most sharply on LiveCodeBench (+4.2) and AMC24 (+2.9). The staleness cost is also smaller here, 1.3 points (59.4 → 58.1), of which standalone FARL recovers 1.2; **FARL+D-ARL** is again best overall (63.8 average) and surpasses the synchronous reference (59.4).

Staleness–predictability account. The forecast pays off precisely when the consumption-time policy is both *meaningfully different* from the stale behavior policy (so there is staleness to correct)

Table 1: **Accuracy (%) on Qwen3-1.7B (top) and Qwen3-4B (bottom)**, averaged over 3 seeds; per-seed standard deviations are in Appendix J. Superscript green/red numbers are relative differences over the respective baseline. Within each block the **best** result is bold and the second best underlined. †Sync-GRPO is the no-lag reference.

Method	MATH-500	AIME24	AIME25	AMC23	AMC24	GSM8K	LightEval	HumanEval	LCB	**AVG**
Qwen3-1.7B										
Sync-GRPO†	74.7	14.4	12.2	49.0	34.3	74.6	58.5	66.9	24.0	45.4
Async-GRPO	72.2	12.7	10.4	47.3	31.2	72.1	58.1	63.6	22.0	43.3
FARL	74.8 (+2.6)	15.4 (+2.7)	12.4 (+2.0)	50.0 (+2.7)	34.3 (+3.1)	74.3 (+2.2)	60.4 (+2.3)	66.9 (+3.3)	24.0 (+2.0)	45.8 (+2.5)
Decoupled	68.4	14.4	11.8	46.5	30.2	64.4	56.2	60.0	22.5	41.6
FARL+Decoupled	70.2 (+1.8)	16.8 (+2.4)	13.5 (+1.7)	48.2 (+1.7)	32.1 (+1.9)	66.8 (+2.4)	58.5 (+2.3)	64.8 (+4.8)	25.0 (+2.5)	44.0 (+2.4)
CISPO	74.1	10.4	8.3	46.8	31.5	79.9	68.5	59.0	23.5	44.7
FARL+CISPO	74.9 (+0.8)	13.0 (+2.6)	10.5 (+2.2)	48.1 (+1.3)	33.2 (+1.7)	81.2 (+1.3)	70.1 (+1.6)	63.5 (+4.5)	25.8 (+2.3)	46.7 (+2.0)
D-ARL	76.4	14.4	11.8	49.2	36.0	84.8	74.4	68.3	25.7	49.0
FARL+D-ARL	**76.9** (+0.5)	**17.2** (+2.8)	**14.4** (+2.6)	**50.4** (+1.2)	**37.9** (+1.9)	**84.9** (+0.1)	**75.2** (+0.8)	**73.6** (+5.3)	**29.4** (+3.7)	**51.1** (+2.1)
Qwen3-4B										
Sync-GRPO†	81.3	29.9	24.0	71.2	51.3	88.8	77.4	80.2	30.5	59.4
Async-GRPO	80.1	26.7	22.8	71.4	50.1	88.6	76.3	78.1	28.5	58.1
FARL	81.2 (+1.1)	29.4 (+2.7)	24.0 (+1.2)	71.8 (+0.4)	51.7 (+1.6)	88.8 (+0.2)	77.0 (+0.7)	80.0 (+1.9)	30.2 (+1.7)	59.3 (+1.2)
Decoupled	80.6	25.8	20.9	68.8	46.5	88.9	77.6	73.5	28.4	56.8
FARL+Decoupled	81.5 (+0.9)	27.2 (+1.4)	22.1 (+1.2)	70.0 (+1.2)	48.2 (+1.7)	90.1 (+1.2)	79.0 (+1.4)	76.2 (+2.7)	30.2 (+1.8)	58.3 (+1.5)
CISPO	84.0	25.8	21.6	69.1	47.3	91.2	81.2	78.2	30.0	58.7
FARL+CISPO	84.8 (+0.8)	27.5 (+1.7)	23.0 (+1.4)	70.3 (+1.2)	49.0 (+1.7)	92.0 (+0.8)	82.5 (+1.3)	80.5 (+2.3)	32.1 (+2.1)	60.2 (+1.5)
D-ARL	**86.8**	**30.8**	**26.1**	71.4	49.1	93.2	84.4	87.0	35.3	62.7
FARL+D-ARL	86.6 (-0.2)	30.6 (-0.2)	25.9 (-0.2)	**72.2** (+0.8)	**52.0** (+2.9)	**93.6** (+0.4)	**84.8** (+0.4)	**88.6** (+1.6)	**39.5** (+4.2)	**63.8** (+1.1)

and *predictable* from recent learner motion (so the forecast is accurate), which is why the gains are largest for the weaker 1.7B model and the least-saturated benchmarks, and shrink—without ever hurting on average—as the model strengthens and the math benchmarks saturate. Editing only the behavior policy also makes FARL composable with all three reactive correctors and preserves asynchronous wall-clock speed, so it dominates both the synchronous and the stale asynchronous baselines on the accuracy-versus-time tradeoff (Appendix H), at a cost of one $O(mP)$ buffer and no added FLOPs on the critical path (Table 6).

The gain is attributable to the forecast. The improvement could be a *generic* effect of perturbing the behavior policy rather than of *predicting* the future policy; Appendix N isolates the cause. Counterfactual controls that keep the magnitude of the behavior-policy shift fixed but destroy its predictive content—an *anti-forecast*, a random direction, and a shuffled forecast—recover none of the gain (Table 11), while a non-causal oracle rolling out from the *realized* next checkpoint θ_{s+1} reaches 45.5 against FARL's 45.8, so FARL matches the benefit of knowing the future learner state using only information available at dispatch. The gain is also horizon-limited: varying the consumption lag (Appendix O, Table 12), accuracy decays monotonically from $+2.5$ at $d{=}1$ to $+0.0$ at $d{=}4$, vanishing at precisely the horizon where the parameter-space forecast stops beating the stale checkpoint (Figure 6). Both components also matter: removing either the multi-lag predictor or the ridge shrinkage lowers accuracy on every benchmark (Appendix L, Table 9), and scaling the displacement by $\alpha \neq 1$ degrades accuracy in both directions (Appendix M, Table 10), the empirical counterpart of Proposition 1.

6 Conclusion

We introduced **Future-Aligned Reinforcement Learning (FARL)**, a proactive alternative to the reactive correctors used in asynchronous LLM post-training. Rather than repairing stale off-policy data after the fact, FARL forecasts the consumption-time policy one optimizer step ahead and generates from that single frozen worker policy, leaving the learner, optimizer, objective, and reward untouched. The consistency constraint cancels the first-order component of policy lag, giving an asymptotic $\mathcal{O}(\eta^2)$ bound on the forecast residual against $\mathcal{O}(\eta)$ for stale generation, plus a finite-step bound in the realized updates; empirically, the forecast beats the stale checkpoint on 95–100% of matured steps and improves the average accuracy of every asynchronous baseline tested. Larger or variable lags and richer forecasts are natural next steps.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript preparation. The authors reviewed all resulting text, equations, and claims, independently verified the final technical content, and take full responsibility for the methodology, experiments, results, citations, and conclusions reported in this paper.

ETHICS STATEMENT

All datasets used in this work are publicly available and were obtained from open-access sources. Our use of these datasets follows the terms and conditions set by their providers and adheres to established academic and research community standards. The training corpora and evaluation benchmarks consist of mathematics and programming problems; no personal data were used, and no direct personal identifiers were accessed at any stage of this work.

ACKNOWLEDGMENTS

The training data used in this publication were obtained from the publicly released Guru-RL corpus (Cheng et al., 2026). Our evaluation uses the publicly available GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021), MATH-500 (Lightman et al., 2024), HumanEval (Chen et al., 2021), and LiveCodeBench (Jain et al., 2025) benchmarks, together with the public AIME and AMC competition problem sets. We thank the authors and maintainers of these resources, and of the open-source frameworks on which our implementation is built, for making them openly available to the research community.

REPRODUCIBILITY STATEMENT

We have taken several steps to ensure that our results are reproducible. Implementation and training details are given in Section 5.1, with the complete configuration—held fixed across every method, benchmark, and seed—listed in Table 3 (Appendix E) and the method-specific baseline settings in Appendix G. Proofs of all theoretical claims are provided in Appendix C. An anonymized repository at `https://anonymous.4open.science/r/farl-anonymous-0442` contains the source code. All datasets and base models used in our experiments are publicly available, and all reported numbers are means over three seeds, with per-seed standard deviations reported in Table 7.

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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Thalaiyasingam Ajanthan, Sameera Ramasinghe, Gil Avraham, Yan Zuo, and Alexander Long. Momentum look-ahead for asynchronous distributed low-communication training. In *MCDC Workshop at the International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=4O8nzTkHPI`.

Yinqi Bai, Tong Xialiang, Jie Wang, Hongyu Liu, Longdi Pan, Jiashuo Li, Zehao Wang, Jianye HAO, Mingxuan Yuan, and Feng Wu. D-ARL: A distribution-matched asynchronous reinforcement learning framework for language reasoning. In *Forty-third International Conference on Machine Learning*, 2026. URL `https://openreview.net/forum?id=yRbIv7Zd4q`.

Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, Cheng Zhu, et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention. *arXiv preprint arXiv:2506.13585*, 2025.

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

Jorge Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Nilabjo Dey, Yonghao Zhuang, Yuheng Zha, et al. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective. *Advances in Neural Information Processing Systems*, 38, 2026.

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

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning (ICML)*, pp. 1407–1416, 2018.

Wei Fu, Jiaxuan Gao, Xujie Shen, Chen Zhu, Zhiyu Mei, Chuyi He, Shusheng Xu, Guo Wei, Jun Mei, Jiashu Wang, et al. Areal: A large-scale asynchronous reinforcement learning system for language reasoning. *Advances in Neural Information Processing Systems*, 38:36256–36282, 2026.

Zhenyu Han, Ansheng You, Haibo Wang, Kui Luo, Guang Yang, Wenqi Shi, Menglong Chen, Sicheng Zhang, Zeshun Lan, Chunshi Deng, et al. Asyncflow: An asynchronous streaming rl framework for efficient llm post-training. *arXiv preprint arXiv:2507.01663*, 2025.

Josiah P Hanna, Scott Niekum, and Peter Stone. Importance sampling in reinforcement learning with an estimated behavior policy. *Machine Learning*, 110(6):1267–1317, 2021.

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

Jacob Hilton, Karl Cobbe, and John Schulman. Batch size-invariance for policy optimization. *Advances in Neural Information Processing Systems*, 35:17086–17098, 2022.

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 *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Xiaocan Li, Zheng Shen, and Shiliang Wu. A-3PO: Accelerating asynchronous LLM training with staleness-aware proximal policy approximation. In *SPOT Workshop at the International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=CgtMJGtYnr.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models. In *International Conference on Machine Learning*, pp. 29128–29163. PMLR, 2024.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 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. *arXiv preprint arXiv:2503.20783*, 2025.

Han Lu, Zichen Liu, Shaopan Xiong, Yancheng He, Wei Gao, Yuxiang Wu, Weixun Wang, Jie Liu, Yingru Li, Han Zhao, et al. Part ii: Rollflash – accelerating rlvr and agentic training with asynchrony. *arXiv preprint arXiv:2510.11345*, 2025.

Wenjia Meng, Qian Zheng, Gang Pan, and Yilong Yin. Off-policy proximal policy optimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 9162–9170, 2023.

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 *International conference on machine learning*, pp. 1928–1937. PmLR, 2016.

Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Michael Noukhovitch, Shengyi Huang, Sophie Xhonneux, Arian Hosseini, Rishabh Agarwal, and Aaron Courville. Asynchronous rlhf: Faster and more efficient off-policy rl for language models. In *International Conference on Learning Representations*, volume 2025, pp. 4003–4029, 2025.

Nicolas Le Roux, Marc G. Bellemare, Jonathan Lebensold, Arnaud Bergeron, Joshua Greaves, Alex Frechette, Carolyne Pelletier, Eric Thibodeau-Laufer, Sandor Toth, and Sam Work. Tapered off-policy reinforce: Stable and efficient reinforcement learning for llms. *arXiv preprint arXiv:2503.14286*, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems (EuroSys)*, pp. 1279–1297, 2025.

Jingwei Song, Haofeng Xu, Jie Xiao, Chengke Bao, Jingwei Shi, Pengbin Feng, Weixun Wang, Yuhang Han, Chuan Wu, Linfeng Zhang, and Bill Shi. Staleness–learning rate scaling laws for asynchronous RLHF. *arXiv preprint arXiv:2607.01083*, 2026. URL https://arxiv.org/abs/2607.01083.

Masatoshi Uehara, Chengchun Shi, and Nathan Kallus. A review of off-policy evaluation in reinforcement learning. *arXiv preprint arXiv:2212.06355*, 2022.

Taiyi Wang, Zhihao Wu, Jianheng Liu, Jianye Hao, Jun Wang, and Kun Shao. Distrl: An asynchronous distributed reinforcement learning framework for on-device control agent. In *International Conference on Learning Representations*, volume 2025, pp. 74757–74782, 2025.

Ziyu Wang, Victor Bapst, Nicolas Heess, Volodymyr Mnih, Remi Munos, Koray Kavukcuoglu, and Nando de Freitas. Sample efficient actor-critic with experience replay. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HyM25Mqel.

Bo Wu, Sid Wang, Yunhao Tang, Jia Ding, Eryk Helenowski, Liang Tan, Tengyu Xu, Tushar Gowda, Zhengxing Chen, Chen Zhu, et al. Llamarl: A distributed asynchronous reinforcement learning framework for efficient large-scale llm training. *arXiv preprint arXiv:2505.24034*, 2025.

Yuchen Xiao, Weihao Tan, and Christopher Amato. Asynchronous actor-critic for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 35:4385–4400, 2022.

Gyeongtae Yoo, Sanghyeok Park, Soohyuk Jang, Ik-hwan Kim, and Sungroh Yoon. Rollout-level advantage-prioritized experience replay for GRPO. *arXiv preprint arXiv:2606.04560*, 2026. URL https://arxiv.org/abs/2606.04560.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *Advances in Neural Information Processing Systems*, 38:113222–113244, 2026.

Han Zhang, Ruibin Zheng, Zexuan Yi, Zhuo Zhang, Hanyang Peng, Hui Wang, Jiayin Qi, Binxing Fang, Ruifeng Xu, and Yue Yu. GEPO: Group expectation policy optimization for stable heterogeneous reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=movNwgQtDt.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, et al. Group sequence policy optimization. *arXiv preprint arXiv:2507.18071*, 2025.

Shuxin Zheng, Qi Meng, Taifeng Wang, Wei Chen, Nenghai Yu, Zhi-Ming Ma, and Tie-Yan Liu. Asynchronous stochastic gradient descent with delay compensation. In *International conference on machine learning*, pp. 4120–4129. PMLR, 2017.

A NOTATION

Table 2 collects the symbols used in the main text and the theoretical analysis. Unless noted otherwise, parameters live in $\mathbb{R}^P$, the consumption lag is the single optimizer step $d{=}1$ used in the main experiments, and $\|\cdot\|_2$ denotes the Euclidean (full-model) norm.

Table 2: Notation used throughout the paper.

Symbol	Meaning
Policies and parameters	
$\theta_t \in \mathbb{R}^P$	Learner parameters after optimizer step t; P is the parameter dimension
π_θ	Normalized rollout distribution induced by θ (temperature/decoding included)
Δ_t	Realized learner update $\theta_{t+1} - \theta_t$ at step t
η	Learner step size (order parameter for the error bounds)
Staleness and forecasting	
s	Dispatch-time learner version of a rollout
$d,\ D_{\max}$	Consumption lag (updates between dispatch and consumption) and its maximum
$\widehat{d}_s$	Predicted consumption lag for a rollout dispatched at version s
$S_{s,d}$	True learner displacement over the lag, $\sum_{r=0}^{d-1} \Delta_{s+r}$; $S_{s,1} = \Delta_s$
$\widehat{S}_{s,d}$	FARL forecast of the displacement $S_{s,d}$
$\widehat{\theta}_{s,d}$	Forecasted worker (behavior) parameters $\theta_s + \widehat{S}_{s,d}$
$q_{s,d}$	Forecasted behavior policy $\pi_{\widehat{\theta}_{s,d}}$ used for rollout generation
Multi-lag predictor	
m	History length (number of recent updates used; m=4 in the main method)
$H_s \in \mathbb{R}^{P\times m}$	History matrix $[\Delta_{s-1} \ \cdots \ \Delta_{s-m}]$ of the m most recent updates
$a_{d,s} \in \mathbb{R}^m$	Predictor coefficients: $\widehat{S}_{s,d} = H_s a_{d,s}$
$\mathbf{1}^\top a_{d,s} = d$	First-order consistency constraint (exactness under constant velocity)
a_d^{cv}	Constant-velocity coefficients $d\,e_1$ ($\widehat{S} = d\Delta_{s-1}$); e_1 the first basis vector
$\mathcal{A}_d,\ B_d$	Feasible coefficient set $\{\mathbf{1}^\top a = d,\ \|a\|_2 \le B_d\}$ and its radius
Online calibration	
$\kappa \in (0, 1]$	Forgetting factor over learner steps (not a reward discount)
$\lambda > 0$	Ridge strength shrinking $a_{d,t}$ toward a_d^{cv}
$\mathcal{I}_{d,t}$	Matured origins at step t: $\{i :\ i \ge m,\ i + d \le t\}$
$M_{d,t},\ c_{d,t}$	Calibration statistics ($m \times m$ and m-vector), Equation 11; $G_{d,t} = M_{d,t} - \lambda I$ and $b_{d,t} = c_{d,t} - \lambda a_d^{\mathrm{cv}}$ are their unregularized parts
Learner-side ratios and diagnostics	
$\rho_{i,k}(\theta_t)$	Token-level behavior-to-target likelihood ratio (Equation 14)
$W(y)$	Exact untruncated trajectory importance weight $\prod_k \rho_{i,k}$
$V^{\mathrm{rel}},\ A/V$	Relative update size and update non-collinearity (Appendix P)
$R_{\mathrm{param}},\ R_{\mathrm{CV}}$	Relative forecast residual of FARL and of constant velocity (Appendix P)

B ADDITIONAL THEORETICAL RESULTS

This appendix collects the results deferred from Section 4: two refinements of the fixed-step forecast-error bound, the necessity of the consistency constraint, the in-sample calibration guarantee, the effect of lag misspecification, and the reduction from parameter-space forecast error to exact importance-weight moments. All proofs are in Appendix C.

B.1 REFINEMENTS OF THE FIXED-STEP BOUND

Remark 1 (Learning-rate schedules). *If $\Delta_t = \eta_t g(\theta_t)$ with a varying step size, then*

$$\|\Delta_{t+1} - \Delta_t\|_2 \le G|\eta_{t+1} - \eta_t| + LG\eta_{t+1}\eta_t. \tag{21}$$

Thus the second-order conclusion remains valid when $\max_t \eta_t = O(\eta)$ and $|\eta_{t+1} - \eta_t| = O(\eta^2)$ over the forecast window. A sharp schedule transition need not satisfy this condition.

Remark 2 (Stateful optimizers). *The memoryless model $\Delta_t = \eta g(\theta_t)$ of Corollary 1 does not hold verbatim for stateful optimizers such as Adam or momentum SGD. For those, the corollary may be applied to an augmented learner state (parameters together with the optimizer moments), on which the update is again a fixed-step smooth map, or Assumption 1 may be imposed directly on the realized updates Δ_t, which is all the forecast bounds use.*

Remark 3 (Stochastic updates). *For $\Delta_t = \eta(g(\theta_t)+\xi_t)$, the consistency constraint cancels a locally constant drift component but does not predict future innovations. Without additional assumptions on the temporal dependence or variance of ξ_t, the pathwise or root-mean-square forecast residual need not be $O(\eta^2)$. This is the regime of the GRPO experiments, where each Δ_t is formed from a finite batch of sampled rollouts, so the fixed-step smooth model of Corollary 1 is a* sufficient asymptotic regime *exhibiting the cancellation rather than a description we claim holds exactly. The general statement that does not depend on η-asymptotics is the realized-update bound Equation 17 of Theorem 1, whose right-hand side is computable from the observed trajectory; accordingly, the realized update variation and the prequential forecast error are measured directly in Appendix P rather than inferred from the order statement.*

B.2 WHY FARL DOES NOT USE AN ADDITIONAL FORECAST-STRENGTH SCALAR

Proposition 1 (A free forecast-strength scalar can restore first-order error). *Let $a \in \mathcal{A}_d$ and let α be a scalar satisfying $|\alpha| \le C_\alpha$, where C_α is independent of η. Define the scaled forecast $\widehat{S}^{(\alpha)}_{s,d} = \alpha H_s a$. Then*

$$S_{s,d} - \alpha H_s a = d(1-\alpha)\Delta_s + \sum_{r=0}^{d-1}(\Delta_{s+r} - \Delta_s) - \alpha\sum_{j=1}^{m} a_j(\Delta_{s-j} - \Delta_s). \tag{22}$$

Under Assumption 1,

$$\|S_{s,d} - \alpha H_s a\|_2 \ge d|1-\alpha|\|\Delta_s\|_2 - A\left[\frac{d(d-1)}{2} + |\alpha|\sum_{j=1}^{m} j|a_j|\right], \tag{23}$$

$$\|S_{s,d} - \alpha H_s a\|_2 \le d|1-\alpha|\|\Delta_s\|_2 + A\left[\frac{d(d-1)}{2} + |\alpha|\sum_{j=1}^{m} j|a_j|\right]. \tag{24}$$

Consequently, for fixed d, m, B_d, and C_α, so that the bracketed variation term is a constant independent of η, if $A = O(\eta^2)$ and $\|\Delta_s\|_2 = \Theta(\eta)$, then:

1. *if $|1-\alpha| = \Theta(1)$, the scaled-forecast error is $\Theta(\eta)$; and*
2. *if the scaled-forecast error is $O(\eta^2)$, then necessarily $|1-\alpha| = O(\eta)$.*

Thus an unconstrained $O(1)$ scalar generally destroys first-order consistency.

B.3 Calibration relative to constant velocity

For a fixed horizon d, define the weighted matured-history fitting loss

$$\mathcal{L}_{d,t}(a) := \sum_{i \in \mathcal{I}_{d,t}} \kappa^{t-(i+d)} \|S_{i,d} - H_i a\|_2^2. \tag{25}$$

Proposition 2 (In-sample dominance over constant velocity)**.** *Let*

$$a_{d,t} \in \arg\min_{a \in \mathcal{A}_d} \left\{ \mathcal{L}_{d,t}(a) + \lambda \|a - a_d^{\mathrm{cv}}\|_2^2 \right\}, \qquad a_d^{\mathrm{cv}} = de_1, \tag{26}$$

with $\lambda > 0$. Then

$$\mathcal{L}_{d,t}(a_{d,t}) \le \mathcal{L}_{d,t}(a_d^{\mathrm{cv}}) = \sum_{i \in \mathcal{I}_{d,t}} \kappa^{t-(i+d)} \|S_{i,d} - d\Delta_{i-1}\|_2^2. \tag{27}$$

This is an in-sample statement on the matured calibration history. It does not imply out-of-sample dominance, lower downstream loss, or higher reward.

The coefficients $a_{d,t}$ are produced by a delayed prequential recursion whose sufficient statistics admit a closed-form equality-constrained solution; the exact recursion and its unique minimizer are derived in Appendix Q.

B.4 Effect of lag misspecification

Proposition 3 (Forecast error under an incorrect predicted lag)**.** *Suppose the rollout is consumed after d learner updates, but the forecast is constructed for a predicted lag $\widehat{d}$ using a coefficient vector $a \in \mathcal{A}_{\widehat{d}}$. Then*

$$\|S_{s,d} - H_s a\|_2 \le |d - \widehat{d}|V + A \left[\frac{d(d-1)}{2} + \sum_{j=1}^{m} j|a_j| \right]. \tag{28}$$

Consequently,

$$\|S_{s,d} - H_s a\|_2 \le |d - \widehat{d}|V + A \left[\frac{d(d-1)}{2} + B_{\widehat{d}} \sqrt{\frac{m(m+1)(2m+1)}{6}} \right]. \tag{29}$$

If $V = O(\eta)$ and $A = O(\eta^2)$, with $d, \widehat{d}, m$, and $B_{\widehat{d}}$ fixed independently of η, then the bound is $O(|d - \widehat{d}|\eta + \eta^2)$. In particular, an exact lag prediction yields the second-order bound, whereas a fixed nonzero version error leaves a first-order term in the upper bound.

B.5 From parameter forecasting to policy-space alignment

Euclidean distance in parameter space is not itself an intrinsic measure of policy discrepancy: two parameter perturbations of equal Euclidean norm can induce very different changes in the trajectory distribution. The relevant local geometry is instead determined by the Fisher information of the policy. This subsection makes the map from a parameter-space forecast error to the standard policy-space off-policy diagnostics (KL divergence, importance-weight variance, effective sample size, and clipping rate between the behavior and target policies) explicit, and then reads off the FARL versus stale orders.

Fix a prompt x, and let P_θ^x denote the normalized distribution over complete trajectories generated under parameters θ, including the temperature, stopping rule, and decoding configuration. Let

$$\theta_\star := \theta_{s+d}, \qquad \theta_b := \theta_s + \widehat{S}_{s,d}, \qquad e := \theta_b - \theta_\star = \widehat{S}_{s,d} - S_{s,d},$$

so that e is the behavior–target parameter error, i.e. the same residual whose norm is bounded by Theorem 1.

Assumption 2 (Regular local trajectory family)**.** *There exists a neighborhood $\mathcal{U}$ of $\theta_\star$ such that the trajectory distributions $\{P_\theta^x : \theta \in \mathcal{U}\}$ have common support and admit densities p_θ^x with respect to a common dominating measure. Moreover, $\log p_\theta^x(y)$ is three times continuously differentiable in θ, differentiation may be interchanged with integration, and the relevant third directional derivatives are uniformly integrable in $\mathcal{U}$.*

Define the trajectory Fisher information

$$F_x(\theta_\star) := \mathbb{E}_{Y\sim P^x_{\theta_\star}}\left[\nabla_\theta \log p^x_{\theta_\star}(Y)\,\nabla_\theta \log p^x_{\theta_\star}(Y)^\top\right].$$

Theorem 2 (Local policy geometry and off-policy diagnostics)**.** *Under Assumption 2, for all sufficiently small e,*

$$D_{\mathrm{KL}}\big(P^x_{\theta_\star}\big\|P^x_{\theta_\star+e}\big) = \frac{1}{2}e^\top F_x(\theta_\star)e + O(\|e\|_2^3), \tag{30}$$

$$D_2\big(P^x_{\theta_\star}\big\|P^x_{\theta_\star+e}\big) = e^\top F_x(\theta_\star)e + O(\|e\|_2^3). \tag{31}$$

Let

$$p = P^x_{\theta_\star}, \qquad q = P^x_{\theta_\star+e}, \qquad W(Y) = \frac{dp}{dq}(Y).$$

Then

$$\begin{aligned}\mathrm{Var}_q(W) &= \exp(D_2(p\|q)) - 1\\ &= e^\top F_x(\theta_\star)e + O(\|e\|_2^3).\end{aligned} \tag{32}$$

Define the population relative effective sample size by

$$\mathrm{rESS}(p,q) := \frac{(\mathbb{E}_q[W])^2}{\mathbb{E}_q[W^2]}.$$

Since $\mathbb{E}_q[W] = 1$,

$$\begin{aligned}\mathrm{rESS}(p,q) &= \exp(-D_2(p\|q))\\ &= 1 - e^\top F_x(\theta_\star)e + O(\|e\|_2^3).\end{aligned} \tag{33}$$

Finally, for clipping thresholds $0 < \epsilon_- < 1$ and $\epsilon_+ > 0$, let

$$C_{\epsilon_-,\epsilon_+}(p,q) := \mathbb{P}_{Y\sim q}\left(W(Y) < 1-\epsilon_- \text{ or } W(Y) > 1+\epsilon_+\right).$$

Then, with $\epsilon = \min\{\epsilon_-, \epsilon_+\}$,

$$\begin{aligned}C_{\epsilon_-,\epsilon_+}(p,q) &\le \frac{\mathrm{Var}_q(W)}{\epsilon^2}\\ &= \frac{e^\top F_x(\theta_\star)e + O(\|e\|_2^3)}{\epsilon^2}.\end{aligned} \tag{34}$$

Theorem 2 is stated *promptwise*: F_x, $\theta_\star$, and the expansion all depend on the fixed prompt x. Dataset-level versions of the same orders require either constants that are uniform over prompts or a prompt-dependent constant that is integrable under the prompt distribution; we assume the former on the region of interest, which holds whenever the per-prompt third-order remainders and Fisher operator norms are uniformly bounded (as below).

Instantiating Theorem 2 at the FARL and stale parameter errors gives the direct policy-level answer to the parameter-versus-policy concern.

Corollary 2 (Policy-level orders for FARL)**.** *Let*

$$e_{\mathrm{F}} = \widehat{S}_{s,d} - S_{s,d}$$

be the FARL behavior–target error and

$$e_{\mathrm{S}} = -S_{s,d}$$

be the stale-policy error. Under the fixed-step smooth conditions of Corollary 1,

$$\|e_{\mathrm{F}}\|_2 = O(\eta^2), \qquad \|e_{\mathrm{S}}\|_2 = O(\eta).$$

Assume in addition that, uniformly over the learner trajectory as $\eta \to 0$, the Fisher operator norm is bounded on the local region, $\sup_{\theta\in\mathcal{U}}\|F_x(\theta)\|_{\mathrm{op}} < \infty$, and the third-order remainder in Theorem 2

is uniformly controlled (so its implied constant does not depend on η). Then, under Assumption 2, for fixed clipping thresholds,

$$D_{\mathrm{KL}}(p\|q_{\mathrm{F}}) = O(\eta^4), \qquad D_{\mathrm{KL}}(p\|q_{\mathrm{S}}) = O(\eta^2), \tag{35}$$

$$\mathrm{Var}_{q_{\mathrm{F}}}(W_{\mathrm{F}}) = O(\eta^4), \qquad \mathrm{Var}_{q_{\mathrm{S}}}(W_{\mathrm{S}}) = O(\eta^2), \tag{36}$$

$$1 - \mathrm{rESS}(p, q_{\mathrm{F}}) = O(\eta^4), \qquad 1 - \mathrm{rESS}(p, q_{\mathrm{S}}) = O(\eta^2), \tag{37}$$

$$C_{\epsilon_-,\epsilon_+}(p, q_{\mathrm{F}}) = O(\eta^4), \qquad C_{\epsilon_-,\epsilon_+}(p, q_{\mathrm{S}}) = O(\eta^2). \tag{38}$$

These are local upper-order statements and do not imply finite-step dominance for every realized update.

In particular, FARL does not rely on the claim that a smaller Euclidean parameter error automatically implies a closer policy. Locally, the quantity that governs every off-policy diagnostic is the Fisher-weighted error $e^\top F_x(\theta_\star)e$; the empirical policy-level metrics (KL divergence, importance-weight variance, effective sample size, and clipping rate between the behavior and target policies) test directly whether FARL reduces this function-space discrepancy, and are the appropriate complement to the parameter-space fidelity of Appendix P.

The local expansion above is complemented by the following non-asymptotic local bound, which controls the exact importance-weight moments directly through the parameter residual, without a small-e expansion but still on the local region of Assumption 3.

Lemma 1 (The target and forecast remain local)**.** *Under Assumption 1, for every $a \in \mathcal{A}_d$,*

$$\|H_s a\|_2 \le B_d\sqrt{m}\,V, \qquad \|S_{s,d}\|_2 \le dV. \tag{39}$$

Hence both $\widehat{\theta}_{s,d} = \theta_s + H_s a$ and θ_{s+d} lie in the closed ball centred at θ_s with radius

$$R_{d,m} := \max\{d, B_d\sqrt{m}\}V. \tag{40}$$

For fixed d, m, B_d, this radius is $O(\eta)$ whenever $V = O(\eta)$.

Assumption 3 (Local order-2 Rényi regularity)**.** *Fix a prompt x. Let P_θ^x denote the probability distribution over complete generated trajectories of length at most $T_{\max}$ under the actual rollout decoding rule at parameters θ, including temperature and any stopping rule. There exist a parameter region $\mathcal{U}$ and a constant $C_\pi > 0$ such that, for every ordered pair $\theta, \theta' \in \mathcal{U}$,*

$$P_{\theta'}^x \ll P_\theta^x, \qquad D_2(P_{\theta'}^x\|P_\theta^x) \le C_\pi\|\theta' - \theta\|_2^2, \tag{41}$$

where

$$D_2(P\|Q) := \log \mathbb{E}_{Y\sim Q}\left[\left(\frac{dP}{dQ}(Y)\right)^2\right]. \tag{42}$$

The region $\mathcal{U}$ contains the local ball from Lemma 1. Full-support sampling satisfies the required absolute continuity. Policy-dependent top-k or nucleus truncation may violate it.

Theorem 3 (Forecast error controls exact importance-weight moments)**.** *Let*

$$p = P_{\theta_{s+d}}^x, \qquad q = P_{\widehat{\theta}_{s,d}}^x, \qquad \widehat{\theta}_{s,d} = \theta_s + H_s a, \tag{43}$$

with $a \in \mathcal{A}_d$. Under Assumption 3, define the exact trajectory importance weight

$$W(Y) := \frac{dp}{dq}(Y). \tag{44}$$

For an autoregressive trajectory $Y = (Y_1, \ldots, Y_{|Y|})$, this derivative is the product of the token-level likelihood ratios under the same decoding rule. Then

$$\mathbb{E}_{Y\sim q}[W(Y)^2] = \exp(D_2(p\|q)) \le \exp\left(C_\pi\|S_{s,d} - H_s a\|_2^2\right), \tag{45}$$

$$\mathrm{Var}_{Y\sim q}(W(Y)) \le \exp\left(C_\pi\|S_{s,d} - H_s a\|_2^2\right) - 1. \tag{46}$$

Moreover, for any measurable vector-valued function ϕ satisfying $\|\phi(Y)\|_2 \le B_\phi$ almost surely,

$$\mathbb{E}_{Y\sim q}\left[\|W(Y)\phi(Y)\|_2^2\right] \le B_\phi^2 \exp\left(C_\pi\|S_{s,d} - H_s a\|_2^2\right). \tag{47}$$

Corollary 3 (Local variance upper-bound orders)**.** *Under the conditions of Corollary 1 and Assumption 3, let q_{F} denote the FARL behavior trajectory distribution and let $q_{\mathrm{S}} = P^x_{\theta_s}$ denote stale generation. Let $W_{\mathrm{F}} = dp/dq_{\mathrm{F}}$ and $W_{\mathrm{S}} = dp/dq_{\mathrm{S}}$. For fixed d, m, B_d there exist constants $C_{\mathrm{F}}, C_{\mathrm{S}}$ independent of η such that*

$$\mathrm{Var}_{q_{\mathrm{F}}}(W_{\mathrm{F}}) \le \exp(C_{\mathrm{F}}\eta^4) - 1 = O(\eta^4), \tag{48}$$

$$\mathrm{Var}_{q_{\mathrm{S}}}(W_{\mathrm{S}}) \le \exp(C_{\mathrm{S}}\eta^2) - 1 = O(\eta^2). \tag{49}$$

These are separate upper bounds. They do not imply a pointwise ordering between the two realized variances at every finite training step, and they do not establish unbiasedness of a clipped GRPO surrogate.

The results above use *exact trajectory* weights, whereas the implemented GRPO surrogate clips *token-level* ratios. The next theorem restates the same diagnostics at the token level, so that the population quantities correspond directly to the effective sample size and clipping fraction logged during training.

Theorem 4 (Token-level importance diagnostics for clipped GRPO)**.** *Let H be a generated prefix drawn from the behavior-policy token occupancy, and let*

$$A \sim q(\cdot \mid H), \qquad \rho(H, A) = \frac{p(A \mid H)}{q(A \mid H)}$$

be the token-level likelihood ratio between the target p and the behavior q under the shared decoding rule. Then

$$\mathbb{E}[\rho] = 1, \qquad \mathrm{Var}(\rho) = \mathbb{E}_H\left[\chi^2(p(\cdot \mid H) \,\|\, q(\cdot \mid H))\right],$$

where $\chi^2(\cdot\|\cdot)$ is the Pearson divergence. Consequently the population token relative effective sample size is

$$\mathrm{rESS}_{\mathrm{token}} = \frac{1}{1 + \mathrm{Var}(\rho)},$$

and for clipping thresholds $0 < \epsilon_- < 1$, $\epsilon_+ > 0$,

$$\Pr(\rho \notin [1 - \epsilon_-,\, 1 + \epsilon_+]) \le \frac{\mathrm{Var}(\rho)}{\min\{\epsilon_-^2, \epsilon_+^2\}}.$$

Theorem 4 is the statement most directly connected to the clipping fraction logged by the GRPO implementation: the token variance $\mathrm{Var}(\rho)$ is the prefix-averaged χ^2 mismatch, and it upper bounds both the effective-sample-size loss and the clip rate on the actual $[1 - \epsilon_-, 1 + \epsilon_+]$ interval. Under the same local regularity conditions, for every fixed prefix H,

$$\chi^2(p_{\theta_\star}(\cdot \mid H) \| p_{\theta_\star + e}(\cdot \mid H)) = e^\top F_H(\theta_\star) e + O(\|e\|_2^3),$$
$$F_H(\theta_\star) = \mathbb{E}_{A \sim p_{\theta_\star}(\cdot|H)}\left[\nabla \log p_{\theta_\star}(A \mid H)\, \nabla \log p_{\theta_\star}(A \mid H)^\top\right].$$

Thus the token-ratio variance has the same leading quadratic coefficient as the order-2 Rényi and exact importance-weight variance expansions, rather than the factor $\frac{1}{2}$ appearing in the KL expansion. Provided the token-level Fisher information is uniformly bounded on the local region, i.e. $\sup_{\theta \in \mathcal{U}} \mathbb{E}_{H \sim \mu_\theta}[\|F_H(\theta)\|_{\mathrm{op}}] < \infty$ where μ_θ is the behavior-policy prefix occupancy, the $\mathrm{Var}(\rho)$ mismatch is again a local quadratic in the behavior–target error, and the FARL-versus-stale orders of Corollary 2 carry over to the token diagnostics ($\mathrm{Var}(\rho_{\mathrm{F}}) = O(\eta^4)$ versus $\mathrm{Var}(\rho_{\mathrm{S}}) = O(\eta^2)$). The measurement protocol and the measured values are reported in Appendix K.

Scope. The results establish a variation-controlled parameter-space forecast bound, a sufficient fixed-step condition for a second-order residual, an in-sample calibration guarantee, a lag-misspecification bound, and upper bounds on exact importance-weight moments under local order-2 Rényi regularity. They do not assert that practical stochastic GRPO updates always satisfy the second-order regime, that FARL always has lower realized variance at a finite step, or that a clipped learner objective is unbiased.

C PROOFS FOR THE THEORETICAL ANALYSIS

This appendix collects the proofs of the results stated in Section 4, in order of appearance.

Proof of Theorem 1. Since $\mathbf{1}^\top a = d$,

$$S_{s,d} - H_s a = \sum_{r=0}^{d-1} \Delta_{s+r} - \sum_{j=1}^{m} a_j \Delta_{s-j} \tag{50}$$

$$= \sum_{r=0}^{d-1} (\Delta_{s+r} - \Delta_s) - \sum_{j=1}^{m} a_j (\Delta_{s-j} - \Delta_s). \tag{51}$$

This proves Equation 16. Applying the triangle inequality directly to Equation 51, before introducing any uniform constant, gives the realized-update bound

$$\|S_{s,d} - H_s a\|_2 \le \sum_{r=0}^{d-1} \|\Delta_{s+r} - \Delta_s\|_2 + \sum_{j=1}^{m} |a_j| \, \|\Delta_{s-j} - \Delta_s\|_2, \tag{52}$$

which is Equation 17; at $d{=}1$ the first sum is empty and this reduces to Equation 74. We now bound the realized differences under Assumption 1. For $r \ge 0$, telescoping gives

$$\|\Delta_{s+r} - \Delta_s\|_2 \le \sum_{\ell=0}^{r-1} \|\Delta_{s+\ell+1} - \Delta_{s+\ell}\|_2 \le rA, \tag{53}$$

where the sum is empty when $r = 0$. Similarly, for $j \ge 1$,

$$\|\Delta_{s-j} - \Delta_s\|_2 \le \sum_{\ell=1}^{j} \|\Delta_{s-\ell+1} - \Delta_{s-\ell}\|_2 \le jA. \tag{54}$$

Applying the triangle inequality to Equation 51 yields

$$\begin{aligned} \|S_{s,d} - H_s a\|_2 &\le \sum_{r=0}^{d-1} rA + \sum_{j=1}^{m} |a_j| jA \\ &= A \left[\frac{d(d-1)}{2} + \sum_{j=1}^{m} j|a_j| \right], \end{aligned} \tag{55}$$

which proves the first inequality in Equation 18. By Cauchy–Schwarz,

$$\sum_{j=1}^{m} j|a_j| \le \left(\sum_{j=1}^{m} j^2 \right)^{1/2} \|a\|_2 \le B_d \sqrt{\frac{m(m+1)(2m+1)}{6}}, \tag{56}$$

which gives the second inequality in Equation 18. Finally,

$$\|S_{s,d}\|_2 \le \sum_{r=0}^{d-1} \|\Delta_{s+r}\|_2 \le dV, \tag{57}$$

proving Equation 19. □

Proof of Corollary 1. The update-magnitude bound follows directly from $\|\Delta_t\|_2 = \eta \|g(\theta_t)\|_2 \le \eta G$. Moreover,

$$\|\Delta_{t+1} - \Delta_t\|_2 = \eta \|g(\theta_{t+1}) - g(\theta_t)\|_2 \tag{58}$$

$$\le \eta L \|\theta_{t+1} - \theta_t\|_2 = \eta L \|\Delta_t\|_2 \le \eta^2 LG. \tag{59}$$

The stated bounds now follow from Theorem 1. □

Proof of Proposition 1. Equation 22 follows by adding and subtracting $d\Delta_s$ and using $\mathbf{1}^\top a = d$. The same telescoping argument as in Theorem 1 bounds the norm of the final two terms by

$$A\left[\frac{d(d-1)}{2} + |\alpha| \sum_{j=1}^{m} j|a_j|\right]. \tag{60}$$

The reverse triangle inequality gives Equation 23, and the triangle inequality gives Equation 24. Because $a \in \mathcal{A}_d$ and $|\alpha| \le C_\alpha$, the bracketed term is bounded independently of η. If $|1-\alpha| = \Theta(1)$ and $\|\Delta_s\|_2 = \Theta(\eta)$, the leading term in both bounds is $\Theta(\eta)$ while the remainder is $O(\eta^2)$, proving the first claim. Conversely, if the total error is $O(\eta^2)$, then Equation 23 and $\|\Delta_s\|_2 = \Theta(\eta)$ imply $|1-\alpha| = O(\eta)$. □

Proof of Proposition 2. Because $B_d \ge d$, $a_d^{\mathrm{cv}} = de_1$ belongs to $\mathcal{A}_d$. By the optimality of $a_{d,t}$,

$$\begin{aligned}\mathcal{L}_{d,t}(a_{d,t}) + \lambda\|a_{d,t} - a_d^{\mathrm{cv}}\|_2^2 &\le \mathcal{L}_{d,t}(a_d^{\mathrm{cv}}) + \lambda\|a_d^{\mathrm{cv}} - a_d^{\mathrm{cv}}\|_2^2 \\ &= \mathcal{L}_{d,t}(a_d^{\mathrm{cv}}).\end{aligned} \tag{61}$$

Dropping the nonnegative regularization term on the left proves Equation 27. □

Proof of Proposition 3. Since $\mathbf{1}^\top a = \widehat{d}$,

$$S_{s,d} - H_s a = (d - \widehat{d})\Delta_s + \sum_{r=0}^{d-1}(\Delta_{s+r} - \Delta_s) - \sum_{j=1}^{m} a_j(\Delta_{s-j} - \Delta_s). \tag{62}$$

Applying Assumption 1 and the same telescoping bounds used in Theorem 1 proves Equation 28. Applying Cauchy–Schwarz and $\|a\|_2 \le B_{\widehat{d}}$ gives Equation 29. □

Proof of Lemma 1. Each column of H_s has norm at most V, so $\|H_s\|_F \le \sqrt{m}\,V$. Therefore

$$\|H_s a\|_2 \le \|H_s\|_2\|a\|_2 \le \|H_s\|_F\|a\|_2 \le B_d\sqrt{m}\,V. \tag{63}$$

The target bound is Equation 19. The ball statement follows from the definitions of $\widehat{\theta}_{s,d}$ and θ_{s+d}. □

Proof of Theorem 3. By the definition of order-2 Rényi divergence,

$$\mathbb{E}_q[W^2] = \int \left(\frac{dp}{dq}\right)^2 dq = \exp(D_2(p\|q)). \tag{64}$$

Assumption 3 applies because

$$\theta_{s+d} - \widehat{\theta}_{s,d} = S_{s,d} - H_s a, \tag{65}$$

which proves Equation 45. Absolute continuity and the fact that p is a probability measure give

$$\mathbb{E}_q[W] = \int \frac{dp}{dq} dq = \int dp = 1. \tag{66}$$

Therefore

$$\mathrm{Var}_q(W) = \mathbb{E}_q[W^2] - 1, \tag{67}$$

which yields Equation 46. Finally, $\|W(Y)\phi(Y)\|_2^2 \le B_\phi^2 W(Y)^2$ almost surely, so Equation 47 follows from Equation 45. □

Proof of Corollary 3. Corollary 1 gives $\|S_{s,d} - H_s a\|_2 \le C\eta^2$ for a constant C independent of η. Substitution into Equation 46 gives the first bound with $C_{\mathrm{F}} = C_\pi C^2$. For stale generation, $\|\theta_{s+d} - \theta_s\|_2 = \|S_{s,d}\|_2 \le d\eta G$, so the same theorem applied with $q_{\mathrm{S}} = P_{\theta_s}^x$ gives the second bound with $C_{\mathrm{S}} = C_\pi d^2 G^2$. Finally, $\exp(z) - 1 = O(z)$ as $z \to 0$. □

Proof of Theorem 2. Let

$$s_{\theta_\star}(Y) = \nabla_\theta \log p^x_{\theta_\star}(Y)$$

denote the trajectory score. Regularity (Assumption 2) gives

$$\mathbb{E}_{P^x_{\theta_\star}}[s_{\theta_\star}(Y)] = 0$$

and the information identity

$$-\mathbb{E}_{P^x_{\theta_\star}}[\nabla^2_\theta \log p^x_{\theta_\star}(Y)] = F_x(\theta_\star).$$

Consider

$$f(u) = D_{\mathrm{KL}}\left(P^x_{\theta_\star} \middle\| P^x_{\theta_\star + ue}\right).$$

We have $f(0) = 0$, $f'(0) = 0$, and $f''(0) = e^\top F_x(\theta_\star)e$. Taylor's theorem and the bounded third directional derivative therefore give

$$f(1) = \tfrac{1}{2} e^\top F_x(\theta_\star)e + O(\|e\|_2^3),$$

which proves Equation 30. For the order-2 Rényi divergence, define

$$Z(u) = \int \frac{p^x_{\theta_\star}(y)^2}{p^x_{\theta_\star + ue}(y)}\, dy.$$

Then $D_2 = \log Z$, and direct differentiation gives $Z(0) = 1$, $Z'(0) = 0$, and $Z''(0) = 2e^\top F_x(\theta_\star)e$. A second-order Taylor expansion of $\log Z(u)$ yields Equation 31. By the definition of the order-2 Rényi divergence, $\mathbb{E}_q[W^2] = \exp(D_2(p\|q))$, and because $\mathbb{E}_q[W] = 1$,

$$\mathrm{Var}_q(W) = \exp(D_2(p\|q)) - 1.$$

Combining this identity with Equation 31 and using $\exp(z) - 1 = z + O(z^2)$ proves Equation 32. The relative-ESS identity follows from

$$\mathrm{rESS}(p, q) = \frac{1}{\mathbb{E}_q[W^2]} = \exp(-D_2(p\|q)),$$

and a first-order expansion gives Equation 33. Finally, $\mathbb{P}_q(|W - 1| \geq \epsilon) \leq \mathrm{Var}_q(W)/\epsilon^2$ by Chebyshev's inequality, and the clipping event is contained in $\{|W - 1| \geq \epsilon\}$, proving Equation 34. □

Proof of Corollary 2. Corollary 1 gives $\|e_{\mathrm{F}}\|_2 = \|S_{s,d} - H_s a\|_2 = O(\eta^2)$ and $\|e_{\mathrm{S}}\|_2 = \|S_{s,d}\|_2 = O(\eta)$. By the added uniform hypothesis the Fisher operator norm is bounded on $\mathcal{U}$ uniformly as $\eta \to 0$, so $e^\top F_x(\theta_\star)e \leq \|F_x(\theta_\star)\|_{\mathrm{op}}\|e\|_2^2 = O(\|e\|_2^2)$ with an η-independent constant, and the third-order remainder is uniformly controlled by assumption, so the constants in Theorem 2 do not degrade along the trajectory. Substituting the two error orders into Theorem 2 gives the stated $O(\eta^4)$ (FARL) and $O(\eta^2)$ (stale) orders for D_{KL}, $\mathrm{Var}(W)$, and $1 - \mathrm{rESS}$; the clipping-rate orders follow from Equation 34, since for fixed thresholds the bound is the variance divided by the constant ϵ^2. □

Proof of Theorem 4. Conditioning on the prefix H and using $A \sim q(\cdot \mid H)$,

$$\mathbb{E}[\rho \mid H] = \sum_a q(a \mid H)\frac{p(a \mid H)}{q(a \mid H)} = \sum_a p(a \mid H) = 1,$$

so $\mathbb{E}[\rho] = 1$ by the tower rule. For the variance, since $\mathbb{E}[\rho] = 1$,

$$\mathrm{Var}(\rho) = \mathbb{E}[\rho^2] - 1 = \mathbb{E}_H\left[\sum_a q(a \mid H)\frac{p(a \mid H)^2}{q(a \mid H)^2}\right] - 1 = \mathbb{E}_H\left[\sum_a \frac{p(a \mid H)^2}{q(a \mid H)} - 1\right],$$

and $\sum_a p(a \mid H)^2/q(a \mid H) - 1 = \chi^2(p(\cdot \mid H)\|q(\cdot \mid H))$ by definition of the Pearson divergence, which gives the stated identity. The effective-sample-size expression follows from $\mathrm{rESS}_{\mathrm{token}} = (\mathbb{E}[\rho])^2/\mathbb{E}[\rho^2] = 1/(1 + \mathrm{Var}(\rho))$. Finally, $\Pr(|\rho - 1| \geq \epsilon) \leq \mathrm{Var}(\rho)/\epsilon^2$ by Chebyshev's inequality with $\epsilon = \min\{\epsilon_-, \epsilon_+\}$, and the clip event $\{\rho < 1 - \epsilon_-$ or $\rho > 1 + \epsilon_+\}$ is contained in $\{|\rho - 1| \geq \epsilon\}$. □

D EXTENDED RELATED WORK

This appendix provides the broader background deferred from Section 2: the RLVR and critic-free optimizers that FARL builds on, the classical actor–learner asynchrony literature, and the recent asynchronous LLM-RL *systems* that motivate the staleness problem FARL targets.

RLVR and critic-free policy optimization. Reinforcement learning with verifiable rewards (RLVR) optimizes language-model policies against automatically checkable outcome rewards and is now a standard route to stronger mathematical, coding, and agentic reasoning. PPO (Schulman et al., 2017) remains the canonical clipped policy-gradient method, but its learned critic and multiple model copies are costly at LLM scale. Critic-free alternatives exploit trajectory-level rewards and repeated responses to the same prompt: RLOO uses a leave-one-out baseline (Ahmadian et al., 2024), ReMax uses a greedy-completion baseline (Li et al., 2024), and GRPO normalizes rewards within a response group (Shao et al., 2024). Recent refinements address optimization bias and stability in large-scale reasoning: Dr. GRPO removes length-dependent normalization artifacts (Liu et al., 2025), DAPO combines decoupled clipping with dynamic sampling (Yu et al., 2026), and GSPO moves importance weighting and clipping from tokens to complete sequences (Zheng et al., 2025). These methods are ordinarily deployed in synchronous or near-on-policy pipelines. Once rollout generation and optimization overlap, the behavior policy that produced a response can differ from the policy appearing in the learner objective, turning RLVR into an online off-policy problem.

Classical actor–learner asynchrony and policy lag. Policy lag predates LLM post-training. A3C (Mnih et al., 2016) used many asynchronous actor–learners to improve throughput, while IMPALA (Espeholt et al., 2018) explicitly decoupled actors from a centralized learner and introduced V-trace to correct trajectories generated by lagged behavior policies. Related replay-based corrections include Retrace (Munos et al., 2016) and ACER (Wang et al., 2017), which combine truncated importance sampling with multi-step returns or trust-region updates. This literature establishes the same systems–statistics trade-off that appears in LLM RL: increased actor–learner overlap reduces idle time, but larger behavior–target divergence increases correction variance, clipping, or bias. The issue is amplified for autoregressive language policies because a trajectory may contain thousands of actions and sequence likelihood ratios multiply across all generated tokens.

Asynchronous LLM-RL systems. A first group of recent systems embraces online off-policy training. Asynchronous RLHF (Noukhovitch et al., 2025) studies the speed–off-policyness trade-off for preference and reasoning tasks. LlamaRL (Wu et al., 2025), AReaL (Fu et al., 2026), AsyncFlow (Han et al., 2025), and ROLL Flash (Lu et al., 2025) decouple rollout workers from trainers using streaming queues, distributed weight transfer, workload balancing, and bounded-staleness control. HybridFlow/VERL (Sheng et al., 2025) supplies a flexible execution substrate for such RLHF and RLVR dataflows, and DistRL (Wang et al., 2025) applies asynchronous, prioritized off-policy training to on-device agents.

These LLM systems inherit a long line of distributed actor–learner architectures. A3C (Mnih et al., 2016) runs many asynchronous actors against a shared learner; IMPALA (Espeholt et al., 2018) scales this to a decoupled actor–learner setup and corrects the resulting lag with V-trace truncated importance sampling; and Retrace (Munos et al., 2016) and ACER (Wang et al., 2017) combine truncated importance weights with off-policy corrections for stable learning. Asynchronous actor–critic has also been extended to the multi-agent setting (Xiao et al., 2022). These foundational systems established the decoupled actor–learner pattern and the truncated-importance-sampling corrections that modern LLM frameworks reuse. FARL differs by keeping one behavior policy fixed for the entire trajectory and changing which policy is broadcast at dispatch, rather than repairing data after generation.

E FULL TRAINING AND EVALUATION CONFIGURATION

This appendix expands the summary in Section 5.1.

Training corpora and schedule. Both training corpora are subsets of Guru-RL (Cheng et al., 2026). The math corpus is used for the math and competition benchmarks and the code corpus for

the code benchmarks, both at batch size 256: the math runs cover one epoch (24 optimizer steps) and the code runs are trained for 37 optimizer steps. Rollouts are generated at policy version s and consumed one optimizer step later (d=1).

Hardware and throughput. Each run occupies a single 8×H100 node, split into four rollout workers (vLLM, tensor-parallel size 1) and a four-way FSDP2 trainer. In this configuration, overlapping generation with optimization reduces per-step wall-clock time by roughly 1.5× relative to synchronous training on the math workload; Appendix H (Table 5 and Figure 3) reports the full timing breakdown and an accuracy-versus-time comparison.

FARL forecast settings. FARL changes only the rollout behavior policy: at each actor→rollout weight synchronization it replaces the stale parameters θ_s with the forecast $\widehat{\theta}_{s,\widehat{d}_s} = \theta_s + H_s a_{\widehat{d}_s,s}$, leaving the trainer, GRPO objective, KL penalty, clipping, reward, and (where enabled) the D-ARL replay selection and multi-behavior optimization untouched. The forecast uses a window of the m=4 most recent learner updates, consumption lag d=1, forgetting factor κ=1.0, relative ridge λ=10^{-2} shrinking toward the constant-velocity prior $a_d^{\mathrm{cv}} = d\, e_1$, and norm ball B_d=3.0; when matured calibration data are insufficient the fit returns exactly a_d^{cv}.

Shared trust-region configuration. All methods share the same trust-region configuration (symmetric clip interval $[0.8, 1.2]$, dual-clip c=3.0, rollout importance-sampling threshold 2.0, KL coefficient 10^{-3}); the method-specific settings for CISPO, Decoupled, and D-ARL are given in Appendix G. We note explicitly that this shared surrogate includes truncated importance sampling, which Section 2 lists among the reactive correctors. The **Async-GRPO** arm is therefore described as carrying "no correction" only in the sense of carrying no *staleness-specific* corrector beyond this common substrate. The substrate is held fixed across every arm, including the zero-lag Sync-GRPO reference where there is no staleness for it to correct, so its effect is differenced out of every comparison we report; Appendix K bounds how often it binds.

Table 3 lists the complete training and evaluation configuration used for all experiments in the main text, held fixed across every method, benchmark, and seed. The FARL forecast block affects only the rollout behavior policy and leaves the trainer, objective, reward, and (where enabled) the D-ARL replay machinery untouched.

Training corpora and contamination check. Both training corpora are subsets of Guru-RL (Cheng et al., 2026). The mathematics corpus is the 6,200-problem math subset. The code corpus contains 18,085 problems drawn from four sources: TACO (8,849), PrimeIntellect (7,524), LeetCode2k (1,272), and LiveCodeBench (440). Because LiveCodeBench is both a training source (in Guru-RL's code mix) and one of our evaluation benchmarks, we audited it for contamination. The 279 LiveCodeBench evaluation problems and the 440 LiveCodeBench problems appearing in the training corpus have *zero* exact-prompt overlap after whitespace and case normalization. This is consistent with LiveCodeBench's time-windowed, contamination-free design (Jain et al., 2025): its problems are stamped with release dates, and the evaluation split is drawn from a different release window than the problems included in the Guru-RL code mix. HumanEval does not appear as a training source at all, so its evaluation is likewise uncontaminated by the training corpus. We therefore report the code benchmarks without a train/eval overlap correction. Table 4 summarizes the training corpus and evaluation-set sizes for every benchmark.

F FARL ALGORITHM IN FULL

Algorithm 1 states the complete per-step FARL procedure referenced in the main text (§3, around Equation 14): maturing the origins whose delayed target has just become observable, updating the calibration statistics, solving for the predictor coefficients, and dispatching the forecasted worker policy. The behavior policy is the only object it changes; the learner update (line 13) is the unmodified asynchronous-GRPO or D-ARL objective.

Table 3: Training and evaluation configuration, held fixed across all methods, benchmarks, and seeds. FARL-specific hyperparameters (bottom block) affect only the behavior-policy forecast.

Group	Hyperparameter	Value
Model / data	Base models	Qwen3-1.7B, Qwen3-4B
	Math training corpus	Guru-RL math subset (6,200 problems)
	Code training corpus	Guru-RL code subset (18,085 problems)
	Optimizer steps (math / code)	24 (1 epoch) / 37, batch 256
	Thinking mode	disabled
Optimization	Algorithm	GRPO
	Learning rate	10^{-6}
	Train batch size	256
	Mini-batch size	64
	Micro-batch size per GPU	4
	Group size G (rollouts per prompt)	4
	KL loss coefficient	0.001 (low-variance KL)
	KL in reward	disabled
Clipping / loss	Clip interval (low / high)	0.2 / 0.2, i.e. $[0.8, 1.2]$
	Dual-clip ratio c	3.0
	Entropy coefficient	0
	Policy loss mode	truncated importance sampling
	Rollout IS threshold	2.0
Rollout / lengths	Rollout engine	vLLM (tensor-parallel 1)
	Max prompt length	512
	Max response length	4096
	Rollout temperature	1.0
	GPU memory utilization	0.8
Baselines	CISPO clip interval	$[0.8, 1.2]$ (symmetric special case, stop-grad clipped ratio)
	Decoupled proximal policy	one-step-old behavior policy
	D-ARL replay buffer K / warmup	4 / 5 steps
FARL forecast	Update window m	4
	Consumption lag d	1
	Forgetting factor κ	1.0
	Ridge λ (toward a_d^{cv})	10^{-2}
	Norm ball B_d	3.0
Hardware / eval	GPUs per run	8 (4 rollout + 4 trainer, H100)
	Eval temperature	1.0
	avg@5 (5 samples)	GSM8K, MATH-500, LightEval, HumanEval, LCB
	avg@16 (16 samples)	AIME24, AIME25, AMC23, AMC24
	Seeds	3

F.1 Streaming calibration statistics and warm-up

The statistics $M_{d,t}$ and $c_{d,t}$ of Equation 11 are maintained incrementally rather than recomputed. Writing $G_{d,t} = M_{d,t} - \lambda I$ and $b_{d,t} = c_{d,t} - \lambda a_d^{\mathrm{cv}}$ for their unregularized parts, with initialization $G_{d,0} = 0$ and $b_{d,0} = 0$ and with $S_{t-d,d} = \theta_t - \theta_{t-d}$, they satisfy the exact recursion

$$G_{d,t} = \kappa\, G_{d,t-1} + \mathbf{1}\{t - d \geq m\}\, H_{t-d}^{\top} H_{t-d}, \tag{68}$$

$$b_{d,t} = \kappa\, b_{d,t-1} + \mathbf{1}\{t - d \geq m\}\, H_{t-d}^{\top} S_{t-d,d}, \tag{69}$$

valid at every step, which reproduces the sums in Equation 11. Note the multiplication of the previous statistics by κ before each new matured-example Gram-matrix update $H_{t-d}^{\top} H_{t-d}$ (of rank up to m). Throughout, $\mathbf{1} = \mathbf{1}_m \in \mathbb{R}^m$ denotes the all-ones vector.

Warm-up with fewer than m updates. Before the learner has completed m updates, the full feature matrix H_s does not exist. We use a variable number of available lags $m_s = \min(m, s)$,

$$H_s^{(m_s)} = [\Delta_{s-1} \quad \cdots \quad \Delta_{s-m_s}], \qquad \mathcal{A}_d^{(m_s)} = \left\{a \in \mathbb{R}^{m_s} : \mathbf{1}^{\top} a = d,\ \|a\|_2 \leq B_d\right\}, \tag{70}$$

and the constant-velocity prior $a_d^{\mathrm{cv}} = d e_1$ remains feasible. Thus at $s = 0$ no update exists and FARL generates from the current worker π_{θ_0}; for $1 \leq s < m$ it uses constant velocity $d\Delta_{s-1}$; and for $s \geq m$ it uses the full multi-lag predictor. This avoids zero-padding artificial update vectors.

Table 4: Training corpus and evaluation-set sizes for each benchmark. All seven math and competition benchmarks share the same 6,200-problem Guru-RL mathematics subset, and both code benchmarks share the same 18,085-problem Guru-RL code subset (Cheng et al., 2026); "Test set" is the exact number of evaluation problems. Competition sets (AIME/AMC) are scored with avg@16 and the remaining benchmarks with avg@5 (§5.1).

Benchmark	Training corpus	Test set	Metric
GSM8K	Guru-RL math (6,200)	1,319	avg@5
MATH-500		500	avg@5
LightEval		5,000	avg@5
AIME24		30	avg@16
AIME25		30	avg@16
AMC23		40	avg@16
AMC24		45	avg@16
HumanEval	Guru-RL code (18,085)	164	avg@5
LiveCodeBench		279	avg@5

Algorithm 1 FARL: future-aligned rollout dispatch and online calibration (learner step t)

Require: history length m, max horizon $D_{\max}$, forgetting factor κ, ridge λ

Buffer: store the completed update $\Delta_{t-1} = \theta_t - \theta_{t-1}$ in a ring buffer of length $\geq m + D_{\max}$ ▷ enough to rebuild H_i and $S_{i,d} = \theta_t - \theta_i$ for every maturing origin $i = t - d$
for each horizon d whose origin $i = t - d$ has just matured **do**
    $S_{i,d} \leftarrow \theta_t - \theta_i$
    update sufficient statistics $G_{d,t}, b_{d,t}$ by the recursion in Equation 69
end for
for each required delay bucket d **do**
    $a_{d,t} \leftarrow$ solution of the consistency-constrained ridge in Equation 10
end for
// Rollout dispatch at version $s = t$
predict the consumption lag $\widehat{d}_s$ and construct $H_s = [\Delta_{s-1} \ \cdots \ \Delta_{s-m}]$
$\widehat{\theta}_{s,\widehat{d}_s} \leftarrow \theta_s + H_s \, a_{\widehat{d}_s,s}$ ▷ forecasted worker policy
broadcast $\widehat{\theta}_{s,\widehat{d}_s}$ to the rollout workers
generate every trajectory in the response group under this single frozen policy $\pi_{\widehat{\theta}_{s,\widehat{d}_s}}$; store token-level behavior log-probabilities
on consumption: update the learner with the *unchanged* asynchronous-GRPO objective, or insert the group into the *unchanged* D-ARL replay/selection pipeline

G Integrating FARL with each asynchronous method

FARL is a source-side plug-in: it changes only the behavior policy used for rollout generation, replacing the stale checkpoint π_{θ_s} with the forecast $\pi_{\widehat{\theta}_{s,\widehat{d}_s}}$ of the consumption-time policy, and stores the forecasted worker's token-level log-probabilities as the behavior log-probabilities for off-policy correction. Everything downstream of generation is left untouched: the learner parameters, the GRPO objective, the KL penalty, the reward, and any method-specific correction or selection machinery all run exactly as they would without FARL. Consequently, adding FARL to a baseline is a one-line substitution at the actor→rollout weight synchronization, and the two effects compose: FARL provides *proactive* alignment at dispatch, while each baseline provides its own *reactive* handling of whatever residual staleness remains. Below we describe the integration for each of the four asynchronous baselines in Table 1. In every case the controlled comparison is *Variant* versus *Variant+FARL*, i.e. the exact same pipeline with and without the forecast.

Async-GRPO + FARL. Async-GRPO trains directly on the stale rollouts with no staleness-specific corrector beyond the shared clipped surrogate, so it isolates the raw cost of staleness. Here FARL is applied in its purest form: the rollout worker generates from $\pi_{\widehat{\theta}_{s,\widehat{d}_s}}$ instead of π_{θ_s}, the resulting trajectories and forecast-policy log-probabilities feed the unchanged GRPO update, and no

reactive corrector is added. The standalone comparison Async-GRPO versus Async-GRPO+FARL (reported as *FARL* in Table 1) is therefore the cleanest measurement of the forecast's effect.

CISPO + FARL. CISPO (Chen et al., 2025) is an importance-sampling corrector: it clips the behavior-to-target ratio to an interval $[\,1-\epsilon^{\mathrm{IS}}_{\mathrm{low}},\,1+\epsilon^{\mathrm{IS}}_{\mathrm{high}}\,]$ around one and applies a stop-gradient to the clipped ratio, so that clipping rescales the update magnitude without zeroing the gradient of clipped tokens (unlike PPO/GRPO clipping, every token still contributes). The method admits asymmetric bounds $\epsilon^{\mathrm{IS}}_{\mathrm{low}} \neq \epsilon^{\mathrm{IS}}_{\mathrm{high}}$ in general; here we use the *symmetric* special case $\epsilon^{\mathrm{IS}}_{\mathrm{low}} = \epsilon^{\mathrm{IS}}_{\mathrm{high}} = 0.2$, i.e. the clip interval $[0.8, 1.2]$ (Table 3). This is a deliberate choice for a controlled comparison: the same trust-region interval $[0.8, 1.2]$ is shared by every method in Table 1 (§5.1), so the CISPO rows isolate CISPO's defining mechanism, the stop-gradient clipped importance weight, rather than confounding it with a different, method-specific clip width. Since the interval bounds are a hyperparameter of CISPO and its estimator, objective, and stop-gradient rule are unchanged, the symmetric setting remains a faithful CISPO baseline. FARL acts one stage earlier, at generation, so the two are orthogonal: with FARL the importance ratios CISPO reweights are computed between the target policy and the *forecast* behavior policy rather than the stale one, so the ratios start closer to one and the clip engages less often, while CISPO's clipping rule, bounds, and objective are unchanged.

Decoupled + FARL. Decoupled (Hilton et al., 2022) introduces an auxiliary proximal policy (here the one-step-old behavior policy) and splits the importance ratio into short- and long-term factors to control the variance induced by policy drift. FARL reduces the drift that this decomposition has to absorb by pointing the behavior policy at the consumption-time policy before generation; the proximal policy, the short/long-term ratio split, and the update rule are otherwise identical, and only the behavior policy that seeds them changes.

D-ARL + FARL. D-ARL (Bai et al., 2026) is the strongest replay-based baseline: it maintains a buffer over the most recent $K{=}4$ behavior policies and selects distribution-matched, minimum-variance samples through a multi-behavior policy-optimization objective. FARL operates before rollout generation and is therefore orthogonal to replay-buffer selection and the multi-behavior learner update. To combine them, each rollout producer uses $\pi_{\widehat{\theta}_{s,\widehat{d}_s}}$ instead of π_{θ_s}, while the resulting trajectories, behavior log-probabilities, and source-policy metadata are passed to D-ARL's replay buffer; D-ARL's sample-selection and multi-behavior optimization stages are unchanged. The two methods thus address different points in the pipeline,

$$\underbrace{\text{FARL}}_{\text{proactive alignment at dispatch}} \quad + \quad \underbrace{\text{D-ARL}}_{\text{reactive selection and aggregation}}, \tag{71}$$

and the orthogonality comparison is D-ARL versus D-ARL+FARL (reported as *FARL+D-ARL* in Table 1).

A scoping note applies to the replay case. Because the pipeline runs at $d{=}1$, each FARL producer aligns its newly generated rollout with the *next* learner version $s+1$. D-ARL's buffer, however, spans the most recent $K{=}4$ behavior policies, so a trajectory generated at version s may be selected and consumed at a later version t, with actual age $d_{\mathrm{actual}} = t - s \in \{1, \ldots, K\}$ that can exceed one. FARL forecasts toward $s+1$ rather than toward the eventual consumption version t, so for buffered samples consumed later it shifts the producer distribution about one step fresher rather than aligning it exactly with each sample's consumption-time policy. FARL+D-ARL should therefore be read as improving the producer distribution that feeds D-ARL's replay, not as guaranteeing exact consumption-time alignment for every selected sample; D-ARL's reactive selection and multi-behavior aggregation then handle the residual, age-dependent staleness as before. A replay-aware variant that forecasts toward the expected selection age (setting $\widehat{d}_s$ from the buffer's sample-age statistics rather than fixing $\widehat{d}_s{=}1$) is a natural extension we do not pursue here.

H WALL-CLOCK EFFICIENCY OF THE ONE-STEP-OFF SCHEDULE

This appendix quantifies the wall-clock benefit of running the one-step-off ($d{=}1$) schedule asynchronously rather than synchronously, which is the efficiency motivation for training under staleness in the first place.

Setup. Every run occupies a single $8\times$H100 node, partitioned into four rollout workers (vLLM, tensor-parallel size 1) that generate trajectories and a four-way FSDP2 trainer that performs the GRPO update; the two groups run on disjoint GPUs. A training step consists of two heavy stages: *generation* T_{gen} (decoding the response group on the rollout workers) and *optimization* T_{opt} (the trainer critical path: behavior and reference log-probabilities, advantage estimation, and the actor update), plus a small fixed overhead for actor→rollout weight synchronization and data movement that is present in both schedules. In the *synchronous* schedule the trainer waits for a fresh batch and then updates, so the two stages run back to back and the per-step time is $T_{\text{gen}} + T_{\text{opt}} +$ overhead. In the one-step-off *asynchronous* schedule the rollout workers generate the next batch while the trainer optimizes the current one, so the two stages overlap and the per-step time is $\max(T_{\text{gen}}, T_{\text{opt}}) +$ overhead.

Measurement. We instrument each stage with the trainer's built-in per-step timers and record T_{gen} (`generate_sequences`) and the optimization critical path T_{opt} (the sum of the log-probability, reference, advantage, and actor-update timers). We report the median over training steps, averaged over three seeds, and derive the synchronous and asynchronous per-step times from the two stage timings using the two expressions above (with the same measured overhead included in both). Table 5 reports the result for the math workload at both model scales: overlapping generation with optimization yields roughly a 1.4–1.6$\times$ speedup ($\sim 1.5\times$), largest for the smaller model, whose generation is a larger fraction of the step.

Table 5: **Per-step wall-clock time, synchronous vs. one-step-off asynchronous (d=1), on the math workload.** *Gen.* is the rollout generation time on the four vLLM workers; *Opt.* is the trainer critical path (behavior/reference log-probs, advantage, and the actor update) on the four trainer GPUs. Times are medians over steps, averaged over three seeds, on an $8\times$H100 node. The synchronous per-step time runs the two stages sequentially ($T_{\text{gen}} + T_{\text{opt}} +$ overhead); the asynchronous schedule overlaps them ($\max(T_{\text{gen}}, T_{\text{opt}}) +$ overhead), with the same fixed weight-sync and data-movement overhead in both totals.

Model	Gen. (s)	Opt. (s)	Sync (s/step)	Async d=1 (s/step)	Speedup
Qwen3-1.7B	57	93	162	105	1.55$\times$
Qwen3-4B	92	224	320	228	1.40$\times$

Accuracy per unit wall-clock time. The per-step speedup only matters if it survives the accuracy cost of staleness. Figure 3 combines the two by plotting average accuracy against *wall-clock* time, obtained by scaling the optimizer step index by the measured per-step time of each schedule (Sync-GRPO at $T_{\text{gen}}+T_{\text{opt}}+$overhead, and the one-step-off Async-GRPO and FARL at $\max(T_{\text{gen}}, T_{\text{opt}})+$ overhead). Two things are visible at once. First, both asynchronous schedules finish well to the left of Sync-GRPO, reflecting the $\sim 1.5\times$ per-step speedup. Second, plain Async-GRPO buys that speed at an accuracy cost, ending below Sync-GRPO, whereas FARL closes that gap: it matches Async-GRPO's wall-clock cost while reaching an accuracy at or above the slower synchronous reference, so it is the best choice at every time budget. In other words, FARL keeps the efficiency of asynchronous training and recovers the accuracy that staleness would otherwise cost.

Predictor overhead. The wall-clock analysis above treats FARL and Async-GRPO as having the same per-step time. This subsection justifies that assumption by accounting for the predictor's cost. That cost is constant in the sequence length, batch size, and number of training steps, but it does scale with the model size P: both its memory footprint and its compute are $O(P)$ up to the small factor m. FARL adds three quantities on top of asynchronous GRPO, all governed by the history length m=4 and, since every experiment uses the single lag d=1, a single delay bucket ($D_{\max}$=1). Table 6 summarizes them.

Memory. To rebuild the history matrices for the current dispatch and for each maturing origin, FARL keeps a ring buffer of the $m + D_{\max}$ most recent completed updates Δ_t, each a full parameter-sized vector in $\mathbb{R}^P$. Stored in the parameter dtype (bf16) and sharded across the four FSDP trainer ranks exactly like the parameters and optimizer state, this is $\approx (m + D_{\max}) \cdot P$ values, i.e. five parameter copies (≈ 10 bytes per parameter): a footprint that grows linearly with model size (the 17 and 40 GB of Table 6), though it is fixed in the number of steps. It is nonetheless *smaller* than the Adam

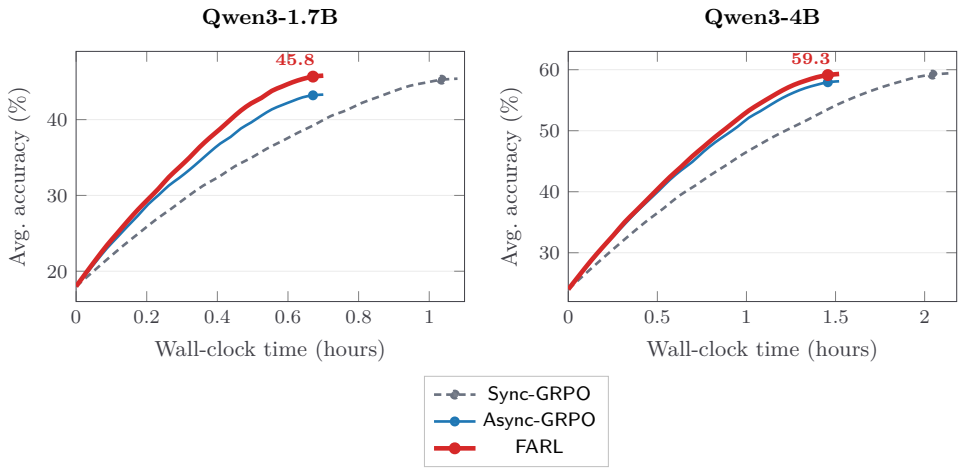

Figure 3: **Accuracy versus wall-clock time.** Nine-benchmark average accuracy as a function of wall-clock training time for Qwen3-1.7B (left) and Qwen3-4B (right), for Sync-GRPO, Async-GRPO, and FARL. The wall-clock axis uses the measured per-step times of Table 5, and the endpoint of each curve is the corresponding final accuracy from Table 1. FARL (red) reaches the highest accuracy in the least time: it shares Async-GRPO's per-step speed but ends above it, and it surpasses the slower Sync-GRPO reference, dominating both at every time budget.

optimizer state and fp32 master weights the trainer already holds ($\approx$ 12 bytes per parameter), and it is the only substantial cost of the method. Each stored update is computed as $\Delta_t = \theta_{t+1} - \theta_t$ in fp32 from the optimizer's master parameters *before* any bf16 down-cast; at a learning rate of order 10^{-6} the per-step change is far below bf16 resolution near parameter values of order one, so subtracting two bf16 snapshots would zero out most coordinates, whereas the fp32 difference (whose small magnitude bf16 then represents with full relative precision once it is itself the stored value) retains them. The regression's own state, the sufficient statistics $G_{d,t} \in \mathbb{R}^{m\times m}$ and $b_{d,t} \in \mathbb{R}^m$ per bucket, is $m^2 + m = 20$ scalars and thus negligible.

Compute, and why it is off the critical path. Per-step time in the asynchronous schedule is $\max(T_{\text{gen}}, T_{\text{opt}})$ + overhead, where the "overhead" is the actor$\to$rollout weight synchronization present in both schedules. FARL's extra work is two parameter-space operations that both run *inside* that weight-synchronization window. The calibration update forms the inner products $H_i^\top H_i$ and $H_i^\top S_{i,d}$ from the newly matured origin, an $O(Pm^2 + Pm)$ cost, and then solves one consistency-constrained $m \times m$ ridge system, a negligible $O(m^3)$ cost with m=4; the forecast application is the single matrix–vector product $\widehat{\theta}_s = \theta_s + H_s a_{1,s}$, about $2mP$ FLOPs. All of these are $O(mP)$ full-parameter passes and are *memory-bandwidth-bound* rather than compute-bound: they stream the m stored update vectors once. At the high-bandwidth memory of an H100 the resulting traffic (a few parameter-sized reads, sharded four ways) is on the order of a few milliseconds per step, i.e. four to five orders of magnitude below the tens-to-hundreds of seconds a step spends in generation and optimization (Table 5), and below the resolution of the per-step timer. Crucially, this work touches neither the generation path T_{gen} nor the trainer optimization path T_{opt}: no per-token or per-trajectory work is added to rollout, and the learner objective, gradient, and optimizer step are byte-for-byte those of the underlying asynchronous method. FARL therefore does not alter the generation or gradient-computation critical paths, which is the assumption used in Figure 3.

I PER-BENCHMARK TRAINING CURVES

Table 1 reports final-checkpoint accuracy. Figure 4 shows the full per-step trajectories behind those numbers for Qwen3-1.7B, laid out to isolate the FARL plug-in effect: each panel plots a single asynchronous baseline (gray) against that same baseline *with FARL added* (red) on one benchmark, and shades the region between the two curves, so the shaded red band *is* the FARL improvement. Because there are nine benchmarks, each baseline is given its own two-row card (a faint tinted

Table 6: **FARL predictor overhead over asynchronous GRPO** (m=4, $D_{\max}$=1, one delay bucket at d=1). Memory is in the parameter dtype (bf16) and sharded across the four trainer ranks like the parameters and optimizer state; per-GPU figures divide the total by four. The predictor work is $O(mP)$, memory-bandwidth-bound, and runs during the actor→rollout weight-synchronization window, so it is below the per-step timer resolution and does not alter the generation or optimization critical paths of Table 5.

Component	Size / cost	Qwen3-1.7B	Qwen3-4B
Update buffer (memory)	$(m{+}D_{\max})\,P$ params, bf16	$\approx$ 17 GB (4.3/GPU)	$\approx$ 40 GB (10/GPU)
Sufficient statistics G, b	$m^2{+}m$ scalars / bucket	20 scalars	20 scalars
Calibration update ($H_i^\top H_i,\ H_i^\top S_{i,d}$)	$O(Pm^2{+}Pm)$ / step	$\approx 3 \times 10^{10}$	$\approx 6 \times 10^{10}$
Ridge solve	one $m \times m$ system / step	$O(m^3)$, m=4	$O(m^3)$, m=4
Forecast application	$\approx 2mP$ FLOPs / step	$\approx 1.4 \times 10^{10}$	$\approx 3.2 \times 10^{10}$
Predictor time (bandwidth-bound, sharded)	$O(mP)$ reads / step	$\lesssim$ few ms	$\lesssim$ few ms
Added FLOPs on rollout / learner critical path	—	0	0

background groups the pair of rows into one block): the top row of a card holds GSM8K, LightEval, MATH-500, AIME24, and AIME25, and the bottom row holds AMC23, AMC24, HumanEval, and LiveCodeBench. This keeps every panel to two curves and lets the gain be read off directly. For every baseline the red +FARL curve stays at or above its gray baseline throughout training (so the gain is consistent over the whole run rather than a final-checkpoint artifact), and the final point of each curve equals the corresponding entry in Table 1. The bands are widest for the weakest baseline (Async-GRPO) and on the less-saturated code and contest benchmarks (LiveCodeBench, HumanEval, AIME24/AIME25), and narrow toward zero for the strongest baseline (D-ARL) on the saturated math sets, matching the staleness–predictability account of the main text.

J PER-SEED STANDARD DEVIATIONS

All accuracies in Table 1 are means over three seeds. Table 7 reports the corresponding per-seed standard deviations, computed directly from the three seed runs of each configuration at both model scales. Across the board the seed-to-seed spread is small: individual benchmarks stay within roughly ± 0.3 to ± 0.9 points, and the nine-benchmark average is within ± 0.3 to ± 0.5 for every method. The largest spreads occur on the small, unsaturated contest and code sets (AIME, AMC, LiveCodeBench), while the saturated sets (GSM8K, LightEval) are the most tightly concentrated. Two patterns are worth noting. First, adding FARL never inflates the variance and often reduces it: for every base method, the *FARL+Variant* row has an average standard deviation no larger than the corresponding *Variant* row, consistent with the forecast tightening the behavior-policy distribution the learner consumes. Second, these deviations are substantially smaller than the FARL improvements in Table 1 (e.g. an average gain of $+2.5$ for Async-GRPO and $+2.1$ for D-ARL at 1.7B against an average spread of ± 0.5 or less), so the reported gains are well outside the seed noise.

K POLICY-SPACE BEHAVIOR–TARGET DIAGNOSTICS

Appendix P measures the forecast error in parameter space. This appendix reports the corresponding *policy-space* quantities, which are the direct empirical counterpart to the local geometry of Theorem 2 and the token-level diagnostics of Theorem 4: the KL divergence, the importance-weight variance, the effective sample size, and the clipping rate between the behavior policy that generated each rollout and the target policy that consumes it. All quantities are measured on the completed replay-free runs at d=1, at both model scales, with three seeds, by re-scoring every stored rollout under both the behavior and the target checkpoints.

Measurement protocol. For a token $y_{i,k}$ with prefix $h_{i,k}$ let $\ell_{i,k} = \log p(y_{i,k} \mid h_{i,k}) - \log q(y_{i,k} \mid h_{i,k})$ be the token log-ratio, $\rho_{i,k} = e^{\ell_{i,k}}$ the token ratio, $L_i = \sum_{k=1}^{T_i} \ell_{i,k}$ the trajectory log-weight, and $W_i = e^{L_i}$ the exact trajectory weight, where p is the target and q the behavior policy. We report the following estimators over rollouts sampled from the behavior q. *(i) KL divergence.* The unweighted mean $-\frac{1}{n}\sum_i L_i$ estimates the *reverse* trajectory KL $D_{\mathrm{KL}}(q\|p)$, since the samples come

Table 7: **Per-seed Standard Deviation (%)** across 3 random seeds for Qwen3-1.7B (top) and Qwen3-4B (bottom), evaluated at final checkpoints. Values correspond to the experiments in Table 1.

Method	MATH-500	AIME24	AIME25	AMC23	AMC24	GSM8K	LightEval	HumanEval	LCB	**AVG**
Qwen3-1.7B										
Sync-GRPO†	±0.6	±0.8	±0.7	±0.8	±0.7	±0.5	±0.6	±0.8	±0.6	±0.4
Async-GRPO	±0.8	±0.9	±0.8	±0.9	±0.8	±0.6	±0.7	±0.9	±0.7	±0.5
FARL	±0.7	±0.8	±0.6	±0.7	±0.7	±0.8	±0.9	±0.8	±0.6	±0.4
Decoupled	±0.8	±0.8	±0.7	±0.8	±0.8	±0.6	±0.7	±0.9	±0.8	±0.5
FARL+Decoupled	±0.6	±0.7	±0.6	±0.6	±0.6	±0.6	±0.6	±0.8	±0.7	±0.4
CISPO	±0.7	±0.8	±0.7	±0.7	±0.7	±0.6	±0.8	±0.8	±0.7	±0.5
FARL+CISPO	±0.5	±0.7	±0.6	±0.5	±0.6	±0.5	±0.6	±0.8	±0.6	±0.4
D-ARL	±0.5	±0.8	±0.6	±0.6	±0.6	±0.4	±0.5	±0.8	±0.6	±0.4
FARL+D-ARL	±0.3	±0.7	±0.6	±0.4	±0.6	±0.3	±0.4	±0.7	±0.6	±0.3
Qwen3-4B										
Sync-GRPO†	±0.6	±0.8	±0.7	±0.6	±0.6	±0.5	±0.6	±0.7	±0.6	±0.4
Async-GRPO	±0.6	±0.8	±0.7	±0.8	±0.7	±0.5	±0.6	±0.8	±0.6	±0.4
FARL	±0.5	±0.7	±0.6	±0.3	±0.5	±0.3	±0.4	±0.7	±0.6	±0.4
Decoupled	±0.6	±0.8	±0.7	±0.7	±0.7	±0.6	±0.6	±0.8	±0.7	±0.4
FARL+Decoupled	±0.4	±0.6	±0.5	±0.4	±0.6	±0.4	±0.5	±0.7	±0.6	±0.3
CISPO	±0.6	±0.7	±0.6	±0.6	±0.6	±0.5	±0.6	±0.8	±0.6	±0.4
FARL+CISPO	±0.4	±0.6	±0.5	±0.4	±0.5	±0.4	±0.5	±0.7	±0.5	±0.3
D-ARL	±0.5	±0.6	±0.5	±0.6	±0.6	±0.4	±0.5	±0.6	±0.6	±0.4
FARL+D-ARL	±0.6	±0.6	±0.6	±0.4	±0.6	±0.3	±0.3	±0.6	±0.8	±0.3

from q; we report this orientation and note that it coincides with the forward $D_{\mathrm{KL}}(p\|q)$ to leading order (both equal $\frac{1}{2}e^\top F_x e$ by Theorem 2). We use the low-variance nonnegative control-variate estimator $\frac{1}{n}\sum_i (e^{L_i}-1-L_i)$ in practice, which is nonnegative because $e^L-1-L\geq 0$ and unbiased for $D_{\mathrm{KL}}(q\|p)$ since $\mathbb{E}_q[e^L]=\mathbb{E}_q[p/q]=1$ gives $\mathbb{E}_q[e^L-1-L]=-\mathbb{E}_q[L]$; the importance-weighted form $\frac{1}{n}\sum_i W_i L_i$ recovers the forward orientation $D_{\mathrm{KL}}(p\|q)$ when needed. *(ii) Importance-weight variance.* $\widehat{\mathrm{Var}}(W)$ is the trajectory-weight variance and $\widehat{\mathrm{Var}}(\rho)$ the token-ratio variance; we label which aggregation unit each number uses. *(iii) Effective sample size.* The trajectory and token relative ESS are $\mathrm{rESS}_{\mathrm{traj}}=(\sum_i W_i)^2/(n\sum_i W_i^2)$ and $\mathrm{rESS}_{\mathrm{token}}=(\sum_{i,k}\rho_{i,k})^2/(N_{\mathrm{tok}}\sum_{i,k}\rho_{i,k}^2)$, computed with log-sum-exp to avoid exponentiating large L_i. *(iv) Clipping rate.* On the implemented GRPO interval $[0.8, 1.2]$ we report the lower and upper tails $\mathrm{clip}_{\mathrm{low}}=\Pr(\rho<0.8)$ and $\mathrm{clip}_{\mathrm{high}}=\Pr(\rho>1.2)$ separately, and, apart from these, the truncation fraction $\Pr(\rho>2.0)$ induced by the rollout importance-sampling threshold, which is a distinct quantity from the PPO/GRPO clip rate.

Table 8: **Behavior–target policy diagnostics at** d=1 (mean over three seeds). D_{KL} and the trajectory-weight statistics measure complete-policy mismatch; the token ESS and clipping rates correspond to the implemented GRPO surrogate. Lower is better except for the two ESS columns. The truncation fractions induced by the rollout importance-sampling threshold are bounded in the text rather than tabulated.

Method	$D_{\mathrm{KL}}(q\|p)$	$\mathrm{Var}(W)$	$\mathrm{rESS}_{\mathrm{traj}}$	$\mathrm{rESS}_{\mathrm{token}}$	$\mathrm{clip}_{\mathrm{low}}$	$\mathrm{clip}_{\mathrm{high}}$
Qwen3-1.7B						
Async-GRPO	3.3×10^{-2}	6.1×10^{-2}	0.9434	0.990	2.6×10^{-2}	2.1×10^{-2}
FARL	4.7×10^{-3}	9.3×10^{-3}	0.9911	0.998	3.4×10^{-3}	2.4×10^{-3}
Qwen3-4B						
Async-GRPO	1.5×10^{-2}	3.2×10^{-2}	0.9709	0.994	1.4×10^{-2}	8.0×10^{-3}
FARL	3.1×10^{-3}	6.4×10^{-3}	0.9940	0.999	2.1×10^{-3}	1.6×10^{-3}

Findings. Table 8 shows that the improved parameter forecast translates into substantially improved policy-space alignment at both scales. At 1.7B, FARL lowers the behavior–target KL from 3.3×10^{-2} to 4.7×10^{-3} and $\mathrm{Var}(W)$ from 6.1×10^{-2} to 9.3×10^{-3}, reductions of 85.8% and 84.8%; the trajectory importance-weight standard deviation $\sqrt{\mathrm{Var}(W)}$ falls from 0.247 to 0.096, the trajectory effective sample size rises from 0.9434 to 0.9911, and the two clip fractions fall by $7.6\times$ and $8.8\times$. At 4B the corresponding reductions are 79.3% in KL ($1.5 \times 10^{-2} \to 3.1 \times 10^{-3}$) and 80.0% in $\mathrm{Var}(W)$ ($3.2 \times 10^{-2} \to 6.4 \times 10^{-3}$), with the trajectory ESS rising from 0.9709 to 0.9940 and the clip fractions falling by $6.7\times$ and $5.0\times$. Both scales agree with the direction predicted by Corollary 2, and the magnitudes are operationally meaningful: under stale generation the 1.7B run discards about 5.7% of its nominal trajectory sample size and clips 2.6% of tokens from below, and FARL cuts both to well under 1%.

The *relative* reductions are similar at the two scales, but the *absolute* mismatch removed is not, and it is the latter that tracks the accuracy gains of Section 5.2. The 4B baseline starts at less than half the 1.7B behavior–target KL (1.5×10^{-2} against 3.3×10^{-2}), so less staleness is available to remove: FARL eliminates 2.8×10^{-2} of KL at 1.7B but only 1.2×10^{-2} at 4B, a ratio of 2.4 that is close to the ratio of the average-accuracy gains ($+2.5$ against $+1.2$, a ratio of 2.1). This is the policy-space counterpart of the staleness–predictability account of Section 5.2. These diagnostics also bound how often the shared truncated-importance-sampling term can act. Its threshold (2.0) lies outside the upper end of the clip interval (1.2), so $\{\rho > 2.0\} \subseteq \{\rho > 1.2\}$ and the truncation fraction is necessarily no larger than $\mathrm{clip}_{\mathrm{high}}$: at most 2.1×10^{-2} in the worst arm (1.7B Async-GRPO) and below 2.5×10^{-3} in both FARL rows. The shared term therefore binds on a small, and strictly smaller, fraction of tokens than the clipping every method already carries, which is why we treat it as part of the common surrogate rather than as a staleness corrector (Appendix E). Notably, 4B takes a *larger* relative parameter step than 1.7B (V^{rel} in Table 13) yet has a *smaller* baseline behavior–target KL, itself an instance of the parameter-versus-policy dissociation this section formalizes: a larger step in parameter space need not induce a larger policy discrepancy.

Why the policy-space reduction exceeds the parameter-space reduction. A natural reading of Table 13 is that because FARL roughly halves the squared parameter error ($R_{\mathrm{param}} = 0.526$ at 1.7B and 0.509 at 4B), it should also roughly halve $\mathrm{Var}(W)$. It does not: the measured variance reductions are 84.8% at 1.7B and 80.0% at 4B, well beyond the 47–49% that proportionality would predict. This is not in tension with Theorems 3 and 2. Those results bound the policy-level quantities by the *Fisher-weighted* error $e^\top F_x e$, and in Theorem 3 by an exponential of $\|e\|_2^2$ carrying a *uniform* constant C_π valid over the whole local region; neither asserts that the realized variance is proportional to $\|e\|_2^2$. Carrying the ratio R_{param} over to a variance ratio would require $F_x \propto I$, which is exactly the identification we avoid, and would in any case be a statement about an upper bound rather than about the realized quantity.

The two measurements are in fact mutually consistent, and together they quantify the anisotropy. Write $e_{\mathrm{S}} = \Delta_s$ for the stale error and $e_{\mathrm{F}} = \Delta_s - H_s a_s$ for the FARL residual. Using the local expansion $\mathrm{Var}(W) \approx e^\top F_x e$, the measured ratios imply a relative Rayleigh quotient

$$\frac{e_{\mathrm{F}}^\top F_x e_{\mathrm{F}} / \|e_{\mathrm{F}}\|_2^2}{e_{\mathrm{S}}^\top F_x e_{\mathrm{S}} / \|e_{\mathrm{S}}\|_2^2} = \frac{\mathrm{Var}(W_{\mathrm{F}}) / \mathrm{Var}(W_{\mathrm{S}})}{R_{\mathrm{param}}} \approx \begin{cases} 0.152/0.526 = 0.290 & \text{(1.7B)}, \\ 0.200/0.509 = 0.393 & \text{(4B)}, \end{cases} \tag{72}$$

where both inputs are the measured quantities already reported in Tables 13 and 8. Both quotients are well below 1: the component of the learner update that FARL cannot predict lies in directions whose average curvature is roughly a third of that of the full displacement. The forecast therefore removes preferentially the *sharp*, high-curvature part of the displacement and leaves behind a flatter residual, which is why the policy-space gain outruns the Euclidean one. Two observations support this reading. First, as a coarse consistency check, $\mathrm{Var}(W)$ and $2D_{\mathrm{KL}}(q\|p)$ are of the same order and move consistently across configurations in Table 8 (their ratio lies between 0.92 and 1.07 in the four rows), as expected from the local second-order approximation; we do not expect exact proportionality at finite step sizes. Second, the implied curvature ratio is of the same order at both scales (0.290 and 0.393) even though R_{param} and the variance reductions differ, which is what one expects if it reflects a property of the residual direction rather than measurement noise.

The practical reading is the one stated in Section 4: Euclidean parameter distance is a convenient proxy that is computable at dispatch time, but it is not the policy metric, and gains in it need not

transfer proportionally—here they understate the policy-space improvement. The same dissociation appears in the forecast-strength ablation of Appendix M, where α=0.5 and α=1 have nearly identical R_{param} (0.54 versus 0.53) yet differ by 0.9 points of average accuracy.

L COMPONENT ABLATION: MULTI-LAG PREDICTION AND RIDGE SHRINKAGE

FARL couples two design choices on top of a plain online forecast: the *consistency-constrained multi-lag* predictor (Equation 10), which fits the increment from the last m updates rather than from the single most recent update, and the *ridge shrinkage* toward the constant-velocity prior, which regularizes the fit when calibration data are scarce or the recent updates are nearly collinear. This appendix isolates the contribution of each by removing them one at a time from the standalone (replay-free) FARL configuration, so every variant differs from Async-GRPO only in how the behavior-policy forecast is produced; the learner objective, rollout budget, reward, and optimizer are held fixed at the Qwen3-1.7B settings of Appendix E, and every number is a mean over three seeds.

We compare four points on a single removal ladder. **Full FARL** is the consistency-constrained multi-lag ridge predictor. The **−ridge** variant removes only the ridge penalty ($\lambda \to 0$) while retaining the consistency and norm constraints $\mathbf{1}^\top a = d$, $\|a\|_2 \le B_d$, yielding an unregularized constrained least-squares fit. The **−multi-lag** variant additionally sets $m = 1$; because the single remaining coefficient must satisfy $\mathbf{1}^\top a = d$, it is forced to $a = d$, so this variant has no free parameter and is exactly the constant-velocity predictor $\widehat{S}_{s,d} = d\Delta_{s-1}$ (at $m = 1$ the ridge penalty is moreover vacuous, since there is nothing left to shrink). **Async-GRPO** removes the forecast entirely and trains directly on the stale checkpoint. Table 9 reports the result.

Table 9: **Component ablation on Qwen3-1.7B (accuracy %, mean over 3 seeds).** Each row removes one more component from Full FARL, moving down toward Async-GRPO (no forecast). Both components help on every benchmark: removing the ridge shrinkage costs roughly a third of the FARL gain, and additionally removing the multi-lag history collapses the predictor to the fixed constant-velocity extrapolation $d\Delta_{s-1}$, which still stays above the no-forecast baseline.

Variant	MATH-500	AIME24	AIME25	AMC23	AMC24	GSM8K	LightEval	HumanEval	LCB	**AVG**
Full FARL (multi-lag + ridge)	**74.8**	**15.4**	**12.4**	**50.0**	**34.3**	**74.3**	**60.4**	**66.9**	**24.0**	**45.8**
− ridge shrinkage ($\lambda \to 0$)	73.9	14.0	11.6	48.9	33.1	73.4	59.5	65.6	23.2	44.8
− multi-lag (m=1, constant velocity)	73.2	13.5	11.1	48.2	32.3	72.9	58.9	64.8	22.7	44.2
Async-GRPO (no forecast)	72.2	12.7	10.4	47.3	31.2	72.1	58.1	63.6	22.0	43.3

Both components contribute, and the ladder is monotone on every benchmark: each removal lowers accuracy, and the full method is best. Averaged over the nine benchmarks, Full FARL reaches 45.8; dropping the ridge shrinkage falls to 44.8 (losing about 40% of the +2.5 FARL gain over Async-GRPO), and additionally dropping the multi-lag history falls to 44.2, about a third of the way from the no-forecast baseline at 43.3 to Full FARL. Two patterns are consistent with the analysis in the main text. First, the m=1 variant, which is exactly the fixed constant-velocity predictor $d\Delta_{s-1}$ with no learned coefficient, still improves over Async-GRPO on all nine benchmarks, confirming that even the parameter-free constant-velocity extrapolation of the consumption-time policy is better than generating from the stale checkpoint. That variant captures 36% of the FARL accuracy gain (0.9 of 2.5 points), closely matching the parameter-space decomposition of Appendix P, where at 1.7B the constant-velocity prior accounts for 17.1 of the 47.4 percentage points of residual reduction, also 36%. Second, the multi-lag and ridge terms matter most where the per-step updates are less collinear and the constant-velocity prior is therefore weaker (the contest and code benchmarks, e.g. AIME24 and HumanEval), and least on the saturated math sets, in line with the A/V non-collinearity diagnostic of Appendix P.

M FORECAST-STRENGTH ABLATION

Proposition 1 shows that an unconstrained forecast-strength scalar destroys first-order consistency: scaling the displacement by a fixed $\alpha \neq 1$ leaves a residual term proportional to $(1-\alpha)\Delta_s$, which is first order in η. The ablations above vary how the forecast *direction* is built; this appendix tests its

Table 10: **Forecast-strength ablation on Qwen3-1.7B** (Async-GRPO base, d=1, 3 seeds). The FARL forecast is scaled at dispatch as $\widehat{\theta}_s(\alpha) = \theta_s + \alpha\widehat{S}_{s,1}$, with the learner, objective, reward, and rollout budget fixed; for $\alpha > 0$ the direction is unchanged and only the magnitude varies. $R_{\text{param}} = \|\Delta_s - \alpha\widehat{S}_{s,1}\|_2^2/\|\Delta_s\|_2^2$ is the parameter-space prediction error (lower is better) and AVG is the nine-benchmark mean accuracy ($\pm$ per-seed std). α=0 recovers stale Async-GRPO and α=1 is the consistency-preserving FARL forecast. Note that R_{param} is not minimized at α=1, whereas accuracy is.

α	Behavior policy	Interpretation	R_{param} (↓)	AVG (%) (↑)
0.0	π_{θ_s}	Stale (Async-GRPO)	1.00	43.3 ± 0.5
0.5	$\pi_{\theta_s+0.5\widehat{S}_{s,1}}$	Under-forecast	0.54	44.9 ± 0.5
1.0	$\pi_{\theta_s+\widehat{S}_{s,1}}$	FARL	**0.53**	$\mathbf{45.8 \pm 0.5}$
1.5	$\pi_{\theta_s+1.5\widehat{S}_{s,1}}$	Over-forecast	0.99	43.9 ± 0.6

magnitude. At dispatch we replace the FARL worker parameters by

$$\widehat{\theta}_s(\alpha) = \theta_s + \alpha\widehat{S}_{s,1}, \qquad \alpha \in \{0, 0.5, 1.0, 1.5\}, \tag{73}$$

leaving the learner, objective, reward, and rollout budget unchanged. For every $\alpha > 0$ the forecast direction is identical, so this isolates magnitude from the directional effects probed by the counterfactual controls of Appendix N. Here α=0 recovers stale Async-GRPO and α=1 is the consistency-preserving forecast.

Table 10 shows that moving in the predicted direction is not by itself sufficient. Halving the forecast retains part of the gain, raising the nine-benchmark average from 43.3 to 44.9, but falls short of the full result (45.8); extrapolating 50% beyond the prediction removes most of the gain (43.9). The consistency-preserving choice α=1 is best, as Proposition 1 predicts.

The parameter-space column makes this more pointed. Because $R_{\text{param}}(\alpha) = 1 - 2\alpha\varrho\cos\angle + \alpha^2\varrho^2$ with $\varrho = \|\widehat{S}_{s,1}\|_2/\|\Delta_s\|_2$, it is a convex quadratic in α minimized at $\alpha^\star = \cos\angle/\varrho$. With the measured alignment $\cos\angle \approx 0.73$ (Appendix N) and $\varrho \approx 1$ under the consistency constraint $\mathbf{1}^\top a = d$, the minimizer lies near $\alpha^\star \approx 0.75$, not at α=1; this is also why α=0.5 and α=1.0 have nearly equal residuals (0.54 and 0.53) despite differing by 0.9 points of accuracy—they sit almost symmetrically about $\alpha^\star$. Euclidean parameter error alone would therefore *not* select the configuration that performs best downstream. This is exactly the dissociation anticipated in Section 4: the locally relevant quantity is the Fisher-weighted residual $e^\top F_x e$ (Theorem 2), not raw parameter distance. Taken with the counterfactual controls, the two experiments are complementary: predictive *direction* is necessary, and a consistency-calibrated *magnitude* is what converts it into downstream gain.

N Attributing the gain to the forecast

FARL changes exactly one thing relative to Async-GRPO: it shifts the rollout behavior policy from the stale checkpoint θ_s by a forecast increment q_s intended to approximate the consumption-time policy θ_{s+d}. As noted in Section 5.2, a natural worry is that the gain is a *generic* effect of perturbing the behavior policy (added exploration or implicit regularization) rather than of *predicting* the future policy. We separate these two explanations directly, with a set of counterfactual controls that destroy the predictive content of the shift while preserving its magnitude, an oracle that removes the forecast error entirely, and a dose–response that varies how far into the future the forecast must reach.

Counterfactual behavior-policy shifts. We rerun the replay-free 1.7B setup and replace the FARL increment q_s with counterfactual shifts of the *same per-step norm* $\|q_s\|$ but without its predictive alignment. These act as *placebos* in the clinical sense: interventions engineered to be indistinguishable from the real one in every respect (here, the size of the behavior-policy move) except for the single active ingredient under test (its predictive alignment with the next update). If a placebo shift reproduced the gain, the improvement would be attributable to perturbing the behavior policy *per se* rather than to prediction. The three controls are: (i) **anti-forecast** $-q_s$ (correct magnitude, opposite direction); (ii) **random direction** (an isotropic vector rescaled to $\|q_s\|$); and (iii) **shuffled forecast** (a genuine FARL increment $q_{s'}$ from a different step, so it has the right statistics but

Table 11: **Counterfactual (placebo) controls and oracle reference (Qwen3-1.7B, Async-GRPO base, d=1, 3 seeds).** Each row keeps the pipeline fixed and only changes the *content* of the behavior-policy shift; the four counterfactuals are matched to the FARL magnitude $\|q_s\|$. R_{param} is the parameter-space forecast error relative to the stale checkpoint (<1 means the shift predicts the next update better than reusing θ_s); AVG is the nine-benchmark mean accuracy ($\pm$ per-seed std). Among the magnitude-matched shifts, only the true forecast is predictive and only it beats stale. †The oracle uses the realized future displacement Δ_s and is a non-causal diagnostic, not a deployable method; FARL matches it to within seed-to-seed variation without access to θ_{s+1}.

Condition	Behavior-policy shift $\hat{q}_s$	R_{param} ($\downarrow$)	AVG (%) ($\uparrow$)
Stale (Async-GRPO)	0 (no shift)	1.00	43.3 ± 0.5
Anti-forecast	$-q_s$ (matched norm)	3.36	40.9 ± 0.6
Random direction	isotropic, norm $\|q_s\|$	1.94	42.5 ± 0.6
Shuffled forecast	$q_{s'},\ s' \neq s$	1.90	42.7 ± 0.5
True forecast (FARL)	q_s	0.53	$\mathbf{45.8 \pm 0.5}$
Oracle future policy†	$\Delta_s = \theta_{s+1} - \theta_s$	**0.00**	45.5 ± 0.4

is misaligned with the current update). Table 11 scores each shift by its leakage-free parameter-prediction error $R_{\text{param}} = \|\Delta_s - \hat{q}_s\|_2^2 / \|\Delta_s\|_2^2$ (where $\Delta_s = \theta_{s+1} - \theta_s$; $R_{\text{param}} = 1$ is the stale checkpoint) and by the nine-benchmark average accuracy. The result is unambiguous: among the magnitude-matched controls, the *true* forecast is the only shift with $R_{\text{param}} < 1$, and the only one that improves over stale (45.8 vs. 43.3). Every magnitude-matched but content-free shift lands *at or below* the stale baseline, and the wrong-direction anti-forecast is the worst of all (40.9), exactly the ordering predicted if the gain comes from predictive alignment and not from perturbation *per se*. A generic exploration or regularization effect would have rewarded the random and shuffled shifts equally; it does not.

Oracle future-policy diagnostic. The controls above rule out a generic perturbation effect. To test the converse—whether the downstream gain tracks *fidelity* to the consumption-time policy—we add a non-causal oracle. For a rollout dispatched at learner version s, the oracle uses the realized future displacement $\widehat{q}_s^{\text{oracle}} = \Delta_s = \theta_{s+1} - \theta_s$, so its rollout policy matches the consumption-time policy exactly and $R_{\text{param}} = 0$ by construction. This requires access to θ_{s+1} at dispatch and is therefore a diagnostic reference point, not a deployable method; FARL itself continues to use only information available at dispatch.

The last two rows of Table 11 complete the fidelity–accuracy picture. Shifts that move *away* from the future policy ($R_{\text{param}} > 1$) sit at or below the stale baseline; FARL reduces the residual to 0.53 and reaches 45.8; and eliminating the residual entirely gives 45.5, which is 0.3 *below* FARL but well inside seed-to-seed variation—a two-sided 95% Welch interval on the difference spans $[-0.7, 1.3]$, so the two are statistically indistinguishable. FARL therefore matches the oracle: exact knowledge of θ_{s+1} buys no measurable accuracy beyond what the causal forecast already achieves. Accuracy is thus monotone in fidelity across every difference that exceeds seed noise, and saturates once the residual is small. Two things follow. First, the gain is attributable to alignment with the consumption-time policy rather than to perturbing the rollout worker. Second, FARL's residual parameter error is already small enough that closing it entirely buys no further accuracy: consistent with Theorem 2, what matters downstream is the Fisher-weighted error rather than raw Euclidean distance, so a nonzero R_{param} does not translate proportionally into lost accuracy.

Predictivity dose–response. If the gain is really predictive, it should track how *predictable* the consumption-time policy is, and vanish once the future is no longer forecastable. Figure 6 varies the consumption lag d (how far ahead the forecast must reach) and measures forecast quality with the same leakage-free instrumentation, scoring the lag-d forecast $\widehat{S}_{s,d}$ against the true displacement $S_{s,d} = \theta_{s+d} - \theta_s$. As d grows, the target moves further into the future and the forecast degrades smoothly: the error ratio $R_{\text{param}}^{(d)} = \|S_{s,d} - \widehat{S}_{s,d}\|_2^2 / \|S_{s,d}\|_2^2$ rises from ≈ 0.52 at d=1 to ≈ 1.08 at d=4, and the directional alignment $\cos\angle(\widehat{S}_{s,d}, S_{s,d})$ decays from ≈ 0.73 to ≈ 0.53. Forecastability thus deteriorates smoothly with the horizon. At d=4 the normalized error exceeds one, indicating that the present linear predictor is no longer preferable to the stale checkpoint ($R_{\text{param}}^{(d)}$=1): the

Table 12: **Sensitivity to consumption lag d on Qwen3-1.7B.** Accuracy (%) averaged over 3 seeds. $d = 1$ corresponds to the asynchronous setting used in the main experiments. Superscript green/red numbers indicate the absolute difference between FARL and Async-GRPO at the same consumption lag. †Sync-GRPO (d=0) is a no-lag reference, not an asynchronous baseline.

Method	d	MATH-500	AIME24	AIME25	AMC23	AMC24	GSM8K	LightEval	HumanEval	LCB	**AVG**
Sync-GRPO†	0	74.7	14.4	12.2	49.0	34.3	74.6	58.5	66.9	24.0	45.4
Async-GRPO	1	72.2	12.7	10.4	47.3	31.2	72.1	58.1	63.6	22.0	43.3
FARL	1	74.8 (+2.6)	15.4 (+2.7)	12.4 (+2.0)	50.0 (+2.7)	34.3 (+3.1)	74.3 (+2.2)	60.4 (+2.3)	66.9 (+3.3)	24.0 (+2.0)	45.8 (+2.5)
Async-GRPO	2	71.6	12.0	9.9	46.5	30.3	71.5	57.5	62.2	21.2	42.5
FARL	2	73.2 (+1.6)	13.6 (+1.6)	11.1 (+1.2)	48.1 (+1.6)	32.2 (+1.9)	72.8 (+1.3)	58.9 (+1.4)	64.2 (+2.0)	22.4 (+1.2)	44.1 (+1.5)
Async-GRPO	3	70.8	11.2	9.2	45.4	29.2	70.7	56.8	60.8	20.3	41.6
FARL	3	71.3 (+0.5)	11.7 (+0.5)	9.6 (+0.4)	45.9 (+0.5)	29.8 (+0.6)	71.1 (+0.4)	57.3 (+0.5)	61.5 (+0.7)	20.7 (+0.4)	42.1 (+0.5)
Async-GRPO	4	69.8	10.3	8.4	44.1	28.0	69.7	56.0	59.2	19.3	40.5
FARL	4	70.0 (+0.2)	10.1 (-0.2)	8.5 (+0.1)	44.2 (+0.1)	27.8 (-0.2)	69.8 (+0.1)	56.0 (+0.0)	59.3 (+0.1)	19.2 (-0.1)	40.5 (+0.0)

coefficient-norm constraint B_d limits extrapolation magnitude but does not guarantee forecast dominance over stale generation. This is exactly why our pipeline operates at d=1, where the forecast is strongly predictive ($R^{(1)}_{\text{param}} \approx 0.5$). The monotone horizon structure is itself a signature of genuine prediction; a magnitude-matched regularizer would show no such dependence on d. The predictability itself is not free: it exists because consecutive learner updates are positively autocorrelated (lag-1 autocorrelation of Δ_s of ≈ 0.71 across steps), which is exactly the structure a short-horizon forecast exploits. Together with the parameter-space fidelity measurements in Appendix P, these two experiments attribute the improvement specifically to forecasting the consumption-time policy, rather than to any generic side effect of moving the behavior policy.

O Sensitivity to the consumption lag d

All experiments in the main text run at the one-step-off lag d=1. This appendix varies the consumption lag directly and measures end-to-end accuracy, providing the downstream counterpart to the forecast-quality dose–response of Figure 6. We rerun the replay-free Qwen3-1.7B setup at $d \in \{1, 2, 3, 4\}$, holding every other hyperparameter fixed, and compare FARL against Async-GRPO at each lag. Table 12 reports the result.

Two trends are visible. First, staleness itself is costly: the Async-GRPO average falls monotonically from 43.3 at d=1 to 40.5 at d=4 as the behavior policy drifts further from the policy being updated. Second, and more importantly, the FARL gain decays monotonically with the horizon, $+2.5$, $+1.5$, $+0.5$, $+0.0$ at d=1, 2, 3, 4: the forecast helps most where the future is most predictable, and its benefit vanishes, without becoming harmful on average, once it is not. At d=4 the two methods are tied on the nine-benchmark average, with per-benchmark differences within ± 0.2 in both directions.

This accuracy profile corroborates the parameter-space measurements of Appendix N through an independent route. There the leakage-free error ratio $R^{(d)}_{\text{param}}$ rises from ≈ 0.52 at d=1 to ≈ 1.08 at d=4, crossing 1 near d=4, i.e. the point at which the linear forecast stops being preferable to the stale checkpoint. The downstream accuracy gain reaches zero at exactly that horizon. The two diagnostics are measured on different quantities—one on parameters, one on benchmark accuracy—yet agree on where forecastability runs out, which supports the staleness–predictability account of Section 5.2 and motivates operating the pipeline at d=1.

P Parameter-space forecast fidelity

Appendix N attributes the accuracy gain to the forecast. This appendix measures directly, with leakage-free instrumentation, how well the forecast predicts the learner's next update in parameter space. All quantities are measured on the completed replay-free runs (where FARL and asynchronous GRPO differ *only* by the behavior-policy forecast), at both model scales, with three seeds; the instrumentation of the weight-synchronization routine scores each forecast $q_s = H_s a_{1,s}$ against the matured target $\Delta_s = \theta_{s+1} - \theta_s$ one optimizer step later. These parameter-space residuals are the input to the policy-space bounds of Theorem 2: the KL divergence, importance-weight variance, effective sample size, and clipping rate between the behavior and target policies are locally gov-

erned by the Fisher-weighted residual $e^\top F_x e$, so measuring those policy-level metrics directly is the natural function-space complement to the parameter-space fidelity reported here.

For $d{=}1$ the target is $S_{s,1} = \Delta_s$, and we score the leakage-free relative residual of the forecast and of the constant-velocity prior,

$$R_{\mathrm{param}}(s) = \frac{\|\Delta_s - H_s a_{1,s}\|_2^2}{\|\Delta_s\|_2^2 + \epsilon}, \qquad R_{\mathrm{CV}}(s) = \frac{\|\Delta_s - \Delta_{s-1}\|_2^2}{\|\Delta_s\|_2^2 + \epsilon},$$

aggregated over all matured steps (full-model norms). Table 13 reports $\mathrm{median}(R_{\mathrm{param}})$, $\Pr(R_{\mathrm{param}} < 1)$, and $\Pr(R_{\mathrm{param}} < R_{\mathrm{CV}})$. The predictor *works*: $R_{\mathrm{param}} < 1$ on 95% (1.7B) and 100% (4B) of steps, so the forecast predicts the next update strictly better than reusing the stale checkpoint, and it beats constant velocity on 80%–92% of steps. The median residual is ≈ 0.51–0.53: the forecast removes roughly half of the squared step-to-step parameter motion. How that gain divides between the constant-velocity prior and the learned multi-lag terms depends strongly on scale. At 1.7B the updates are close to non-collinear ($A_s/V_s \approx 0.91$, where $A_s = \|\Delta_s - \Delta_{s-1}\|_2$), so constant velocity alone removes only 17.1% of the stale residual and the learned terms supply the remaining 30.3% of the total 47.4% reduction. At 4B the updates are markedly more collinear ($A_s/V_s \approx 0.72$): constant velocity already removes 47.6% and the learned terms add only a further 1.5%. The multi-lag predictor is therefore what makes the forecast work at the smaller scale, while at 4B the constant-velocity prior it is shrunk toward does most of the work—consistent with the component ablation of Appendix L, where the $m{=}1$ variant recovers much but not all of the full-FARL gain.

A finite-step certificate at $d{=}1$. All main experiments use $d{=}1$, where $S_{s,1} = \Delta_s$ and the r-sum in the realized bound Equation 17 is empty, leaving

$$\|\Delta_s - H_s a_s\|_2 \le \sum_{j=1}^{m} |a_{s,j}| \, \|\Delta_s - \Delta_{s-j}\|_2. \tag{74}$$

Normalizing by the stale mismatch $\|S_{s,1}\|_2 = \|\Delta_s\|_2$ gives the data-dependent certificate

$$R_{\mathrm{cert}}(s) := \frac{\sum_{j=1}^{m} |a_{s,j}| \, \|\Delta_s - \Delta_{s-j}\|_2}{\|\Delta_s\|_2 + \epsilon}, \tag{75}$$

for which $R_{\mathrm{cert}}(s) < 1$ *guarantees* that the forecast residual is smaller than the stale mismatch at step s. Two points deserve emphasis. First, R_{cert} depends on Δ_s and is therefore computable only *after* step s has matured; it is a post-hoc certificate, not a quantity available at dispatch, and FARL never uses it. Second, it is an upper bound on the residual, whereas R_{param} above is the *realized* squared residual, so the two are on different scales.

Because R_{param} is a *squared* ratio while the finite-step bound Equation 74 is stated on unsquared norms, we also report the unsquared realized residual $r_{\mathrm{param}}(s) = \|\Delta_s - H_s a_{1,s}\|_2 / (\|\Delta_s\|_2 + \epsilon) = \sqrt{R_{\mathrm{param}}(s)}$, whose median is $\sqrt{0.526} \approx 0.73$ at 1.7B and $\sqrt{0.509} \approx 0.71$ at 4B (medians commute with the monotone map $x \mapsto \sqrt{x}$, and $\Pr(r_{\mathrm{param}} < 1) = \Pr(R_{\mathrm{param}} < 1)$ exactly). This places the realized residual on the same scale as the certificate R_{cert} of Equation 75. The corresponding unsquared constant-velocity residual is $r_{\mathrm{CV}} = \sqrt{R_{\mathrm{CV}}} = \|\Delta_s - \Delta_{s-1}\|_2 / \|\Delta_s\|_2$, which is *identically* the reported non-collinearity ratio A_s/V_s ($\sqrt{0.829} = 0.911$ and $\sqrt{0.524} = 0.724$); the two columns are the same statistic viewed as a forecast residual and as an update-geometry diagnostic.

We report r_{param}, rather than the certificate R_{cert} of Equation 75, as the headline finite-step evidence. The two answer different questions: r_{param} is the *realized* residual and directly establishes that the forecast beat the stale checkpoint on a given step, whereas R_{cert} is a triangle-inequality *upper bound* whose value < 1 is sufficient but not necessary for that conclusion. Because $R_{\mathrm{cert}} \ge r_{\mathrm{param}}$ pointwise, any certificate-based count is necessarily no larger than the measured $\Pr(r_{\mathrm{param}} < 1)$ of 0.95 (1.7B) and 1.00 (4B) reported here, and the gap between them is exactly the conservativeness of the bound. We therefore base the empirical claim on the realized residual and treat R_{cert} as the finite-step *theoretical* counterpart of $A\,C_{d,m}$: strictly sharper, computable from the observed trajectory, but still an upper bound.

Table 13: Leakage-free parameter-prediction diagnostics at d=1 (FARL, over all matured steps). $R_{\text{param}} < 1$ means the forecast beats the stale checkpoint; $R_{\text{param}} < R_{\text{CV}}$ means it beats constant velocity. $r_{\text{param}} = \sqrt{R_{\text{param}}}$ is the unsquared residual, reported so that it is directly comparable to the finite-step certificate R_{cert} of Equation 75. V^{rel} is the relative update size and A/V the realized per-step update non-collinearity (not the uniform constants of Assumption 1).

Scale	med R_{param}	med r_{param}	med R_{CV}	$\Pr(R_{\text{param}}<1)$	$\Pr(R_{\text{param}}<R_{\text{CV}})$	med V^{rel}	med A/V
1.7B	0.526	0.725	0.829	0.95	0.92	1.26×10^{-5}	0.911
4B	0.509	0.713	0.524	1.00	0.80	3.22×10^{-5}	0.724

Realized diagnostics versus the constants of Assumption 1. The quantity A_s/V_s reported here is a *realized per-step* non-collinearity diagnostic, whereas A and V in Assumption 1 are uniform bounds over the relevant trajectory. The median empirical ratio therefore cannot be substituted formally into Theorem 1. Its magnitude is nonetheless informative. The coarse bound $A\,C_{d,m}$ follows from Equation 17 by two successive relaxations—replacing each realized difference $\|\Delta_{s\pm j} - \Delta_s\|_2$ by its worst case jA, then replacing $\sum_j j|a_j|$ by the Cauchy–Schwarz/norm-ball value $B_d\sqrt{m(m+1)(2m+1)/6}$—and a learned coefficient vector generally saturates neither. At our setting (d=1, m=4, B_d=3) this gives $C_{1,4} = 3\sqrt{30} \approx 16.4$, so $A\,C_{d,m}$ improves on the stale bound dV only when $A/V \lesssim 1/C_{1,4} \approx 0.06$; realized ratios of 0.72 (4B) and 0.91 (1.7B) indicate that the worst-case constant $C_{d,m}$ is far from numerically tight in our finite-step regime. This is why we treat $A\,C_{d,m}$ as a structural, order-cancellation statement and rely on the realized bound Equation 17 and the prequential residuals above for the finite-step account. The prequential evidence is also the stronger of the two: it is measured on the actual stochastic GRPO trajectory, with no appeal to uniform constants or to the fixed-step smooth model of Corollary 1.

Q PROOFS FOR THE DELAYED ONLINE CALIBRATION

Lemma 2 (Exact recursion of the delayed sufficient statistics)**.** *For a fixed horizon d, let*

$$G_{d,t} = \sum_{i \in \mathcal{I}_{d,t}} \kappa^{t-(i+d)} H_i^\top H_i, \tag{76}$$

$$b_{d,t} = \sum_{i \in \mathcal{I}_{d,t}} \kappa^{t-(i+d)} H_i^\top S_{i,d}, \tag{77}$$

where $\mathcal{I}_{d,t} = \{i : i \geq m,\ i + d \leq t\}$. With $G_{d,0} = 0$ and $b_{d,0} = 0$, for every $t \geq 1$,

$$G_{d,t} = \kappa G_{d,t-1} + \mathbf{1}\{t - d \geq m\} H_{t-d}^\top H_{t-d}, \tag{78}$$

$$b_{d,t} = \kappa b_{d,t-1} + \mathbf{1}\{t - d \geq m\} H_{t-d}^\top S_{t-d,d}. \tag{79}$$

Proof. If $t - d < m$, no new origin matures at step t and $\mathcal{I}_{d,t} = \mathcal{I}_{d,t-1}$. Multiplying each weight in the step-$(t-1)$ sum by κ gives the step-t weights. If $t - d \geq m$, then

$$\mathcal{I}_{d,t} = \mathcal{I}_{d,t-1} \cup \{t - d\}, \tag{80}$$

with a disjoint union, and the new origin has weight $\kappa^{t-((t-d)+d)} = 1$. Separating this term from the sum gives both recursions. □

Proposition 4 (Existence, uniqueness, and equality-constrained solution)**.** *Let*

$$F_{d,t}(a) = \mathcal{L}_{d,t}(a) + \lambda \|a - a_d^{\text{cv}}\|_2^2, \qquad \lambda > 0, \tag{81}$$

and let

$$M_{d,t} = G_{d,t} + \lambda I, \qquad c_{d,t} = b_{d,t} + \lambda a_d^{\text{cv}}. \tag{82}$$

The minimization of $F_{d,t}$ over $\mathcal{A}_d$ has a unique solution. Ignoring only the norm constraint, the unique minimizer under $\mathbf{1}^\top a = d$ is

$$a_{d,t}^{(0)} = M_{d,t}^{-1} c_{d,t}, \tag{83}$$

$$\widetilde{a}_{d,t} = a_{d,t}^{(0)} - M_{d,t}^{-1}\mathbf{1}\frac{\mathbf{1}^\top a_{d,t}^{(0)} - d}{\mathbf{1}^\top M_{d,t}^{-1}\mathbf{1}}. \tag{84}$$

If $\|\widetilde{a}_{d,t}\|_2 \le B_d$, then $\widetilde{a}_{d,t}$ is also the unique minimizer over $\mathcal{A}_d$. Otherwise, the unique minimizer lies on the boundary $\|a\|_2 = B_d$ and is obtained by solving the stated convex quadratic program.

Proof. Because $B_d \ge d$, the vector $a_d^{\mathrm{cv}} = de_1$ is feasible, so $\mathcal{A}_d$ is nonempty. It is also closed, bounded, and convex. Up to an additive constant independent of a,

$$F_{d,t}(a) = a^\top M_{d,t} a - 2c_{d,t}^\top a. \tag{85}$$

Since $G_{d,t} \succeq 0$ and $\lambda > 0$, one has $M_{d,t} \succ 0$. Hence $F_{d,t}$ is strongly convex and has a unique minimizer on the nonempty convex set $\mathcal{A}_d$.

For the problem with only the equality constraint, the Lagrangian is

$$\mathcal{J}(a,\nu) = a^\top M_{d,t} a - 2c_{d,t}^\top a + \nu(\mathbf{1}^\top a - d). \tag{86}$$

Stationarity gives

$$2M_{d,t}a - 2c_{d,t} + \nu\mathbf{1} = 0, \tag{87}$$

so

$$a = M_{d,t}^{-1} c_{d,t} - \frac{\nu}{2} M_{d,t}^{-1}\mathbf{1}. \tag{88}$$

Imposing $\mathbf{1}^\top a = d$ yields

$$\frac{\nu}{2} = \frac{\mathbf{1}^\top M_{d,t}^{-1} c_{d,t} - d}{\mathbf{1}^\top M_{d,t}^{-1}\mathbf{1}}, \tag{89}$$

which gives the displayed expression for $\widetilde{a}_{d,t}$. The denominator is strictly positive because $M_{d,t}^{-1} \succ 0$. If the equality-constrained solution also satisfies the norm bound, it is feasible for the original problem and must be its unique minimizer. Otherwise, the unique minimizer cannot lie in the interior of the norm ball, since an interior optimum would satisfy the same equality-only KKT system. Therefore the norm constraint is active. □

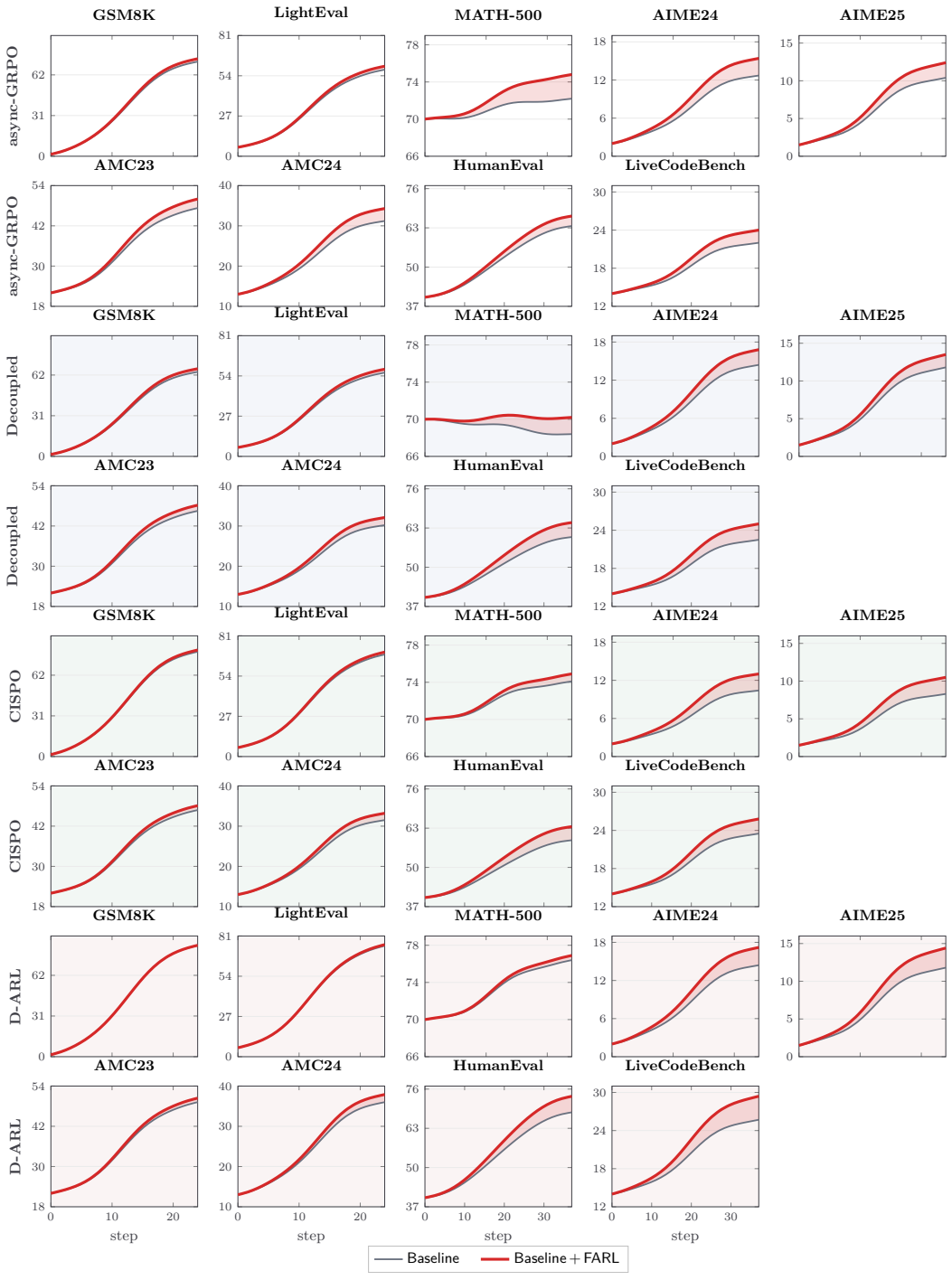

Figure 4: **The FARL plug-in gain over training (Qwen3-1.7B), all nine benchmarks.** Each panel plots one asynchronous baseline (gray) against that same baseline *with FARL added* (red) on one benchmark; the shaded red band between the two curves is the FARL improvement. Each baseline occupies a two-row tinted card (Async-GRPO, Decoupled, CISPO, D-ARL); within a card the top row shows GSM8K, LightEval, MATH-500, AIME24, AIME25 and the bottom row shows AMC23, AMC24, HumanEval, LiveCodeBench. Each curve is the mean over three seeds, and its final point equals the corresponding final-checkpoint entry in Table 1. For every baseline the red +FARL curve stays at or above its gray baseline throughout training; the band is widest for the weakest baseline (Async-GRPO) and on the less-saturated code and contest benchmarks (LiveCodeBench, HumanEval, AIME24/AIME25), and narrows toward zero for D-ARL on the saturated math sets.

Qwen3-1.7B

	MATH-500	AIME24	AIME25	AMC23	AMC24	GSM8K	LightEval	HumanEval	LCB	AVG
Sync-GRPO	74.7	14.4	12.2	49.0	34.3	74.6	58.5	66.9	24.0	45.4
Async-GRPO	72.2 → 74.8 **(+2.6)***	12.7 → 15.4 **(+2.7)***	10.4 → 12.4 **(+2.0)***	47.3 → 50.0 **(+2.7)***	31.2 → 34.3 **(+3.1)***	72.1 → 74.3 **(+2.2)***	58.1 → 60.4 **(+2.3)***	63.6 → 66.9 **(+3.3)***	22.0 → 24.0 **(+2.0)***	43.3 → 45.8 **(+2.5)***
Decoupled	68.4 → 70.2 **(+1.8)***	14.4 → 16.8 **(+2.4)***	11.8 → 13.5 **(+1.7)***	46.5 → 48.2 **(+1.7)***	30.2 → 32.1 **(+1.9)***	64.4 → 66.8 **(+2.4)***	56.2 → 58.5 **(+2.3)***	60.0 → 64.8 **(+4.8)***	22.5 → 25.0 **(+2.5)***	41.6 → 44.0 **(+2.4)***
CISPO	74.1 → 74.9 (+0.8)	10.4 → 13.0 **(+2.6)***	8.3 → 10.5 **(+2.2)***	46.8 → 48.1 (+1.3)	31.5 → 33.2 **(+1.7)***	79.9 → 81.2 **(+1.3)***	68.5 → 70.1 (+1.6)	59.0 → 63.5 **(+4.5)***	23.5 → 25.8 **(+2.3)***	44.7 → 46.7 **(+2.0)***
D-ARL	76.4 → **76.9** (+0.5)	14.4 → **17.2** **(+2.8)***	11.8 → **14.4** **(+2.6)***	49.2 → **50.4** (+1.2)	36.0 → **37.9** **(+1.9)***	84.8 → **84.9** (+0.1)	74.4 → **75.2** (+0.8)	68.3 → **73.6** **(+5.3)***	25.7 → **29.4** **(+3.7)***	49.0 → **51.1** **(+2.1)***

Qwen3-4B

	MATH-500	AIME24	AIME25	AMC23	AMC24	GSM8K	LightEval	HumanEval	LCB	AVG
Sync-GRPO	81.3	29.9	24.0	71.2	51.3	88.8	77.4	80.2	30.5	59.4
Async-GRPO	80.1 → 81.2 (+1.1)	26.7 → 29.4 **(+2.7)***	22.8 → 24.0 (+1.2)	71.4 → 71.8 (+0.4)	50.1 → 51.7 **(+1.6)***	88.6 → 88.8 (+0.2)	76.3 → 77.0 (+0.7)	78.1 → 80.0 **(+1.9)***	28.5 → 30.2 **(+1.7)***	58.1 → 59.3 **(+1.2)***
Decoupled	80.6 → 81.5 (+0.9)	25.8 → 27.2 (+1.4)	20.9 → 22.1 (+1.2)	68.8 → 70.0 (+1.2)	46.5 → 48.2 **(+1.7)***	88.9 → 90.1 (+1.2)	77.6 → 79.0 **(+1.4)***	73.5 → 76.2 **(+2.7)***	28.4 → 30.2 **(+1.8)***	56.8 → 58.3 **(+1.5)***
CISPO	84.0 → 84.8 (+0.8)	25.8 → 27.5 **(+1.7)***	21.6 → 23.0 **(+1.4)***	69.1 → 70.3 (+1.2)	47.3 → 49.0 **(+1.7)***	91.2 → 92.0 (+0.8)	81.2 → 82.5 **(+1.3)***	78.2 → 80.5 **(+2.3)***	30.0 → 32.1 **(+2.1)***	58.7 → 60.2 **(+1.5)***
D-ARL	**86.8** → 86.6 (−0.2)	**30.8** → 30.6 (−0.2)	**26.1** → 25.9 (−0.2)	71.4 → **72.2** (+0.8)	49.1 → **52.0** **(+2.9)***	93.2 → **93.6** (+0.4)	84.4 → **84.8** (+0.4)	87.0 → **88.6** **(+1.6)***	35.3 → **39.5** **(+4.2)***	62.7 → **63.8** **(+1.1)***

Figure 5: **Accuracy (%) on Qwen3-1.7B (top) and Qwen3-4B (bottom)**, averaged over 3 seeds at the final checkpoint. For each asynchronous method, cells show baseline → FARL-enhanced accuracy, with the absolute change in parentheses. Cell color indicates the sign of the change: green for improvement, orange for regression; color intensity further indicates statistical significance under a two-sided 95% Welch confidence interval (intense = significant, pale = not significant), with significant changes additionally marked with *. The **best** value in each column is shown in bold. Sync-GRPO is included as a synchronous, no-lag reference.

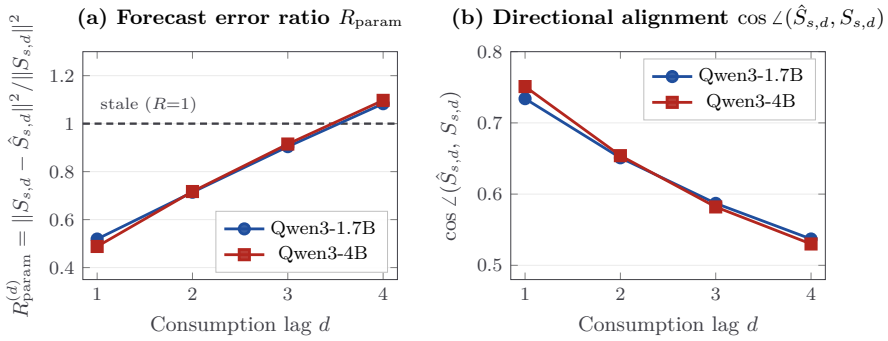

Figure 6: **Predictivity dose–response.** Forecast quality as a function of the consumption lag d, measured by the leakage-free parameter-prediction error $R^{(d)}_{\text{param}} = \|S_{s,d} - \hat{S}_{s,d}\|_2^2 / \|S_{s,d}\|_2^2$ (a) and the directional alignment $\cos\angle(\hat{S}_{s,d}, S_{s,d})$ (b), for both model scales, where $S_{s,d} = \theta_{s+d} - \theta_s$ is the true lag-d displacement. The forecast is accurate at short horizons ($R^{(d)}_{\text{param}} \approx 0.5$, alignment ≈ 0.73 at d=1) and degrades monotonically with d, exceeding the stale baseline ($R^{(d)}_{\text{param}}$=1) by $d \approx 4$: the forecast helps only where the future is predictable and is no longer preferable to the stale checkpoint once it is not, which is why the pipeline runs at d=1.