SKIP WHAT YOU CAN PREDICT: PREDICTIVE REPOSITIONING FOR POLICY OPTIMIZATION FOR EFFICIENT LLM TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Reinforcement learning with verifiable rewards (RLVR) can improve the reasoning ability of large language models, but repeatedly updating a policy on the same rollout batch is expensive: every additional update requires another backward pass, and multi-step methods pay for all intermediate optimization steps. We introduce Predictive Repositioning for Policy Optimization (PrePO), which uses two observed optimizer transitions to estimate a farther point along the same-batch optimization trajectory, moves partway toward that point, evaluates the original objective there, and applies a corrective update. This gives PrePO a fixed active cost that does not grow with the virtual optimization depth. Our analysis gives finite-horizon error bounds for AdamW and Muon and shows that sufficiently accurate endpoint estimates preserve the usual descent and convergence behavior of smooth gradient descent. In RLVR experiments, PrePO reaches matched performance targets in fewer optimization steps and lower wall-clock time than the corresponding baselines. We further evaluate the same update mechanism in supervised fine-tuning on a different dataset, showing that its use is not restricted to the original RLVR setting. Together, these results suggest that PrePO provides a practical mechanism for approximating repeated same-batch optimization while avoiding the cost of explicitly executing every intermediate update.

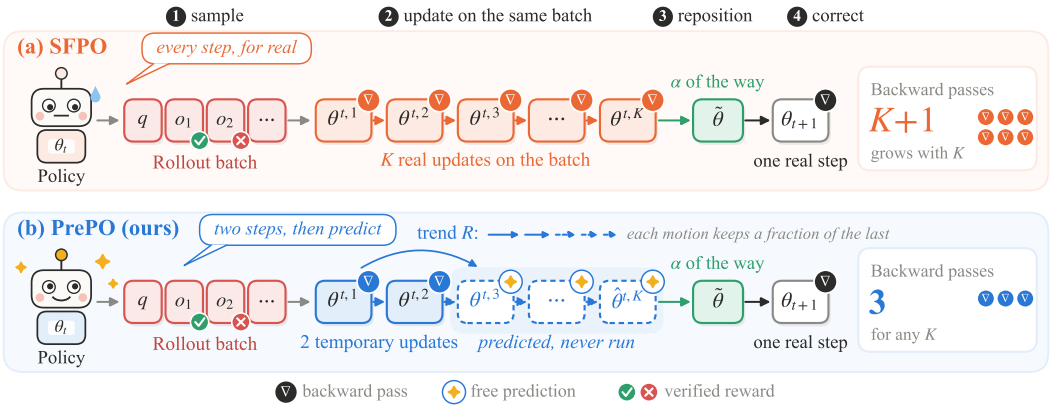

Figure 1: SFPO and PrePO on the same rollout batch. (a) SFPO runs all K updates for real before it repositions and corrects, so its cost grows with K. (b) PrePO runs two temporary updates, predicts the rest of the path from how the motion is changing, moves a fraction α toward the predicted endpoint, and takes one real step there, three backward passes for any K.

1 INTRODUCTION

Reinforcement learning with verifiable rewards (RLVR) improves an LLM by repeatedly updating a policy from sampled responses and their verified rewards. The rollout, reward, and advantage

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.

computations are already costly, and every policy update adds another backward pass. In practice, one rollout batch is often reused for several updates, as in the multiple epochs of PPO (Schulman et al., 2017), so the policy follows a local trajectory before the next batch is collected. Reaching a target accuracy therefore depends not only on the quality of the learning signal, but also on how many updates are executed on each batch.

Several RL optimizers use more than one update from the same batch. GRPO learns from group-relative rewards without a separate critic (Shao et al., 2024), while SFPO executes several policy updates on one rollout batch before applying a correction (Wang et al., 2026b). Classical ideas such as extragradient, vector extrapolation, and Lookahead also use trial points, extrapolated iterates, or fast inner updates (Korpelevich, 1976; Sidi et al., 1986; Zhang et al., 2019). These methods can move farther along a local trajectory, but the intermediate updates are real optimizer steps, so their backward-pass cost grows with the number of steps. This raises a basic question for RLVR: **can the trajectory information be retained without executing every intermediate policy update?**

We address this question with Predictive Repositioning for Policy Optimization (PrePO, Figure 1). PrePO is developed first for RLVR: it uses two temporary policy transitions to estimate where a farther update would land, moves partway toward that estimate, and lets the original policy objective determine the committed update. The data, reward construction, loss, and optimizer remain unchanged; only the execution of the update sequence changes. The active path uses three optimizer updates, independent of the virtual depth K. After establishing the RLVR result, we test whether the same schedule transfers to supervised fine-tuning (SFT) and on-policy distillation (OPD), and we evaluate Muon (Jordan, 2024) as an alternative optimizer.

2 Related Work

Our primary setting is reinforcement learning with verifiable rewards (RLVR), where policy-gradient methods improve a model from sampled responses and their rewards. GRPO learns from group-relative rewards without a separate critic (Shao et al., 2024; Mroueh, 2025; Mroueh et al., 2025); DAPO changes clipping and dynamic sampling for large-scale training (Yu et al., 2025); GSPO moves importance-ratio computation and clipping to the sequence level (Zheng et al., 2025a); CAPO uses curvature information to identify unstable token updates (Melo et al., 2026); and VAPO develops a value-based augmented PPO framework for long-chain reasoning (Yue et al., 2025). Other work reduces rollout or systems cost through selective sampling, adaptive allocation, distributed orchestration, or speculative decoding (Zheng et al., 2025b; Wu et al., 2026; Jiang et al., 2026; Xu et al., 2026; Sheng et al., 2025; He et al., 2025; Kim et al., 2026). These methods modify the learning signal, data budget, or training pipeline. PrePO starts after the rollout batch is fixed: it takes two ordinary updates as temporary probes, predicts how the same-batch motion would continue, moves partway toward that predicted point, and keeps one corrective update.

Our transfer analysis also covers supervised fine-tuning (Wei et al., 2022; Ouyang et al., 2022) and on-policy distillation (Hinton et al., 2015; Gu et al., 2024; Agarwal et al., 2024). There too PrePO leaves the data, parameterization, and objective fixed and changes only how the objective is executed on a batch.

Extrapolation in optimization. Table 1 sorts classical extrapolation methods by how far ahead they look and how they get there. Extragradient, mirror-prox, and optimistic methods (Korpelevich, 1976; Nemirovski, 2004; Popov, 1980; Rakhlin & Sridharan, 2013; Daskalakis et al., 2018; Gidel et al., 2019; Mertikopoulos et al., 2019; Mokhtari et al., 2020), momentum (Polyak, 1964; Nesterov, 1983), and gradient extrapolation (Lan & Zhou, 2018; Lin et al., 2020) look one step ahead. Vector extrapolation and Anderson acceleration look to where an iteration finally settles, using a stored history of iterates (Richardson, 1911; Aitken, 1927; Cabay & Jackson, 1976; Sidi et al., 1986; Anderson, 1965; Walker & Ni, 2011; Scieur et al., 2016). Lookahead, policy mirror descent with lookahead, and SFPO reach a point K steps ahead by taking every step (Zhang et al., 2019; Protopapas & Barakat, 2024; Wang et al., 2026b). Weight prediction looks a small fixed number of steps s ahead by assuming the current update repeats unchanged (Chen et al., 2018; Guan et al., 2023), like predicting where a car will be from its speed alone.

PrePO predicts from the speed *and how it is changing*. Two probe steps on the batch show, for each coordinate, whether the optimizer's motion is holding its pace or slowing down, and PrePO

Table 1: Where each extrapolation family aims and how it gets there. A fixed point is where a repeated iteration stops moving, which a one-update RLVR batch never reaches; s is a small fixed horizon such as a pipeline delay.

Family	Aims at	How the lookahead point is obtained	Tracks motion change	Cost fixed in K	Backward passes
Extragradient, optimistic, momentum	1 step	a trial step, the previous gradient, or past iterates	–	✓	1–2 per step
Vector extrapolation, Anderson	fixed point	least-squares fit over a stored history of iterates	✓	×	1 per iterate, to convergence
Weight prediction	s steps	repeat the current update unchanged	×	✓	1 per step
Lookahead, SFPO	K steps	execute all K updates	✓	×	K or $K{+}1$
Checkpoint extrapolation	later run	extend a line through saved checkpoints	×	✓	none, no correction
PrePO (ours)	K steps	fit the motion trend from two probes, project it K steps	✓	✓	**3 for any K**

carries that trend K steps forward for three backward passes and checks it with a fresh corrective gradient. Repeating the current update, like a larger learning rate, stretches every coordinate by the same amount, including the ones going back and forth, so on sharp coordinates its step grows (Proposition 2); with $R = I$ PrePO reduces to this constant-step prediction. Other work predicts the training run rather than the batch, carrying weights forward along a nearly straight line through saved checkpoints (Zheng et al., 2024; Cai et al., 2026; Wang et al., 2026a). PrePO works inside a single batch, where the trend is measured afresh each time and every predicted point is followed by a real gradient step, so the two act at different scales.

3 Background: RL Updates and Their Extensions

In RLVR, the current policy samples a rollout batch B_{RL} containing prompts, responses, rewards, and the quantities needed by the policy objective. Once this batch is fixed, GRPO or another policy optimizer evaluates the same objective at successive parameter states. SFT and OPD expose the same update interface with different batches and losses: B_{SFT} is a supervised minibatch, while B_{OPD} contains the inputs and teacher targets used by the on-policy distillation loss.

Let $\mathcal{T} \in \{\mathrm{RL}, \mathrm{SFT}, \mathrm{OPD}\}$ index the training setting, let $\mathcal{L}_{\mathcal{T}}(\theta; B_{\mathcal{T}})$ be its objective, and let s^j be the state of a possibly stateful optimizer. At update j, the objective gradient and optimizer transition are

$$g_{\mathcal{T}}^j = \nabla_\theta \mathcal{L}_{\mathcal{T}}(\theta^j; B_{\mathcal{T}}), \qquad (\theta^{j+1}, s^{j+1}) = \mathcal{U}_{\mathcal{T}}(\theta^j, s^j; g_{\mathcal{T}}^j), \tag{1}$$

where $\mathcal{U}_{\mathcal{T}}$ includes the chosen optimizer and its state. The observed parameter displacement is $\Delta^j = \theta^{j+1} - \theta^j$. Reusing the same batch for several updates produces the local trajectory $\theta^0 \to \theta^1 \to \cdots \to \theta^K$. Under plain gradient descent, this trajectory satisfies

$$\theta^K = \theta^0 - \eta \sum_{j=0}^{K-1} g_{\mathcal{T}}^j, \qquad \theta^1_{\mathrm{large}} = \theta^0 - \eta' g_{\mathcal{T}}^0, \tag{2}$$

where η is the usual step size and η' is a larger one. A single large step rescales one gradient direction. Repeated small steps recompute $g_{\mathcal{T}}^j$ at each new point, and a stateful optimizer also updates its state, so the final displacement reflects both accumulated magnitude and a changing direction. Neither shortcut gives this move for free. A larger learning rate scales every coordinate alike, even those whose motion is reversing, and more epochs per batch pay one backward pass per update. PrePO uses these observed transitions and does not require a closed-form model of the optimizer. In the following sections, we omit the subscript $\mathcal{T}$ when the training setting is clear.

4 MOTIVATION: PREDICT THE ENDPOINT, NOT EVERY INTERMEDIATE STEP

The intermediate states along this path are expensive because each optimizer step needs a backward pass, yet they do not collect new data; they mainly reveal how the optimizer moves on the current batch. PrePO treats this path as a sequence of optimizer motions z^j (an AdamW direction, a Muon matrix displacement, or an SGD step) and asks how each motion turns into the next. Two probe steps answer this per coordinate. A trend $r \approx 1$ means the coordinate keeps its pace, and $0 < r < 1$ means it is slowing down. Carrying these trends forward gives the K-step destination without taking the remaining steps, rather than assuming that every coordinate keeps its latest pace. Defining the residual $e^j = z^{j+1} - Rz^j$ makes $z^{j+1} = Rz^j + e^j$ an identity rather than a model. The endpoint error is therefore an explicit sum of residuals, bounded by quantities logged during training (Section 6), and no quadratic objective or frozen Hessian is assumed. The prediction only selects where the true objective is evaluated again; the committed update always uses a fresh gradient.

5 PREPO: PREDICT, REPOSITION, THEN CORRECT

PrePO starts from a simple observation: when several optimizer updates reuse the same batch, the first two updates already reveal how the parameters are moving and how that movement is changing. From these two updates alone, PrePO predicts where K updates would have ended, moves partway toward that predicted point, and evaluates the objective once there. Only the resulting corrective update is kept, so an active PrePO step costs three backward passes regardless of K. The procedure has the same four stages for every optimizer—only the definition of *motion* (the displacement produced by one update) and *motion trend* (the rate at which successive motions change) differ. We describe each stage in general terms below, then show the concrete forms for AdamW and Muon; formal derivations and error bounds appear in Appendices A–B.

5.1 TEMPORARY UPDATES: LEARN THE LOCAL DIRECTION OF TRAVEL

Why two updates and not one? One optimizer step shows where the parameters moved, but not whether that movement is speeding up or slowing down. A second step on the same batch shows exactly that, so two steps are the minimum that reveals the local *motion trend*. PrePO reuses the fixed batch for two temporary updates:

$$(\theta^t, s^t) \xrightarrow{\mathcal{O}(B)} (\theta^{t,1}, s^{t,1}) \xrightarrow{\mathcal{O}(B)} (\theta^{t,2}, s^{t,2}). \tag{3}$$

Each update produces a *motion*, and the change between the two motions is the motion trend. What counts as a "motion" depends on the optimizer: for AdamW it is the adaptive per-coordinate step, for Muon it is the realized matrix displacement, and for SGD it is the negative gradient step. In every case, both updates are temporary: PrePO saves the original parameters and optimizer state beforehand and restores them afterward, so the probes measure how the optimizer is moving without contributing any training progress.

5.2 PREDICT: TURN TWO OBSERVED MOTIONS INTO A VIRTUAL K-STEP ENDPOINT

The two temporary updates give two motions on the same batch, z_0 and z_1. Inside one PrePO step we drop the iteration index and write $\theta_0, \theta_1, \theta_2$ for $\theta^t, \theta^{t,1}, \theta^{t,2}$. If the motion keeps changing the same way, where would K updates on this batch end? To answer it, PrePO describes the change from z_0 to z_1 with a motion trend R, chosen so that $Rz_0 \approx z_1$. It helps to think of R as a speed ratio. A ratio near one means the motion keeps its pace, a ratio between zero and one means it is slowing down, and a negative ratio means it is going back and forth. Each optimizer measures this ratio at its own level. AdamW sets the step of every parameter separately, so every parameter gets its own ratio $r_i = z_{1,i}/z_{0,i}$. Muon updates a whole weight matrix at once, so every matrix gets a single ratio ρ. Once the trend is known, PrePO can estimate the motions it never computes: the third is about Rz_1, the fourth about $R^2 z_1$, and so on. These estimates are free, because they reuse only the two motions already measured. A motion is not yet a change in the parameters, so PrePO applies each estimated motion the same way the optimizer applies a real one, $\theta_{n+1} = C\theta_n + Az_n$. Here A turns a motion into a parameter change and C carries the current parameters forward. For AdamW the motion is the adaptive direction, so $A = -\eta I$ applies the learning rate and $C = I - \eta\lambda I$ applies

Algorithm 1: One PrePO step on a fixed batch B. Left: the general procedure. Right: what its terms are for AdamW and Muon. Both loops use the same update rule $\theta_{n+1} = C\theta_n + Az_n$: the probe loop runs it with real gradients, and the predict loop replays it with predicted motions. The probes are discarded; only the last update is kept.

General procedure (any optimizer)
Input: parameters θ_0, state s_0, depth K, blend α; batch gradient $g(\theta) = \text{clip}(\nabla \mathcal{L}_B(\theta))$

1. Probe *(2 real updates)*
- Save (θ_0, s_0)
- **for** $n = 0, 1$ **do**
  - Real step with $g(\theta_n)$: $\theta_n \to \theta_{n+1}$
  - Record its motion z_n (right panel)

2. Predict *(no gradients)*
- Fit the trend R so that $Rz_0 \approx z_1$
- *// start at the first probe (k=1 gives θ_2)*
- $\hat{\theta} \leftarrow \theta_1,\ z \leftarrow z_1$
- **for** $k = 1$ **to** $K - 1$ **do**
  - Replay one step: $\hat{\theta} \leftarrow C\hat{\theta} + Az$
  - Next motion: $z \leftarrow Rz$
- *// $\hat{\theta}$ is the predicted endpoint*

3. Correct *(1 real update)*
- Move partway: $\tilde{\theta} \leftarrow \theta_0 + \alpha(\hat{\theta} - \theta_0)$
- Real step from s_0: $(\theta, s) \leftarrow \mathcal{O}(\tilde{\theta}, s_0, g(\tilde{\theta}))$

What the symbols mean

Every step has the form $\theta_{n+1} = C\theta_n + Az_n$: z_n is the *motion*, C carries the old parameters forward, and A scales the motion (Section 5.2).

AdamW

z_n	Adam direction $d_n = \hat{m}_n/(\sqrt{\hat{v}_n} + \epsilon)$
$C,\ A$	$(1 - \eta\lambda)I,\ -\eta I$
R	$\text{diag}(r_i),\ r_i = \text{clip}(z_{1,i}/z_{0,i})$

Muon

z_n	whole matrix change $\Theta_{n+1} - \Theta_n$
$C,\ A$	$I,\ I$
R	$\rho I,\ \rho = \text{clip}\big(\langle z_0, z_1\rangle_F / \|z_0\|_F^2\big)$

$\hat{m}_n, \hat{v}_n$: bias-corrected Adam moments; $\langle \cdot, \cdot \rangle_F$: Frobenius inner product. Clip ranges: Appendix A.3; fallback mechanisms: Appendix C.6.

weight decay. For Muon the motion is already the full matrix change, so $A = C = I$. Starting from θ_1, PrePO applies this rule $K - 1$ times and calls the result the predicted endpoint $\hat{\theta}$, or $\hat{\theta}^{t,K}$ when the iteration matters. For Muon this is simply the first temporary update plus the estimated motions, $\hat{\theta} = \theta_1 + \sum_{q=0}^{K-2} \rho^q z_1$. Setting $R = I$ repeats the latest motion, which is weight prediction. The fitted trend costs nothing beyond the two probes. Where motion keeps its pace it gives the same endpoint, and where motion slows or reverses it moves that parameter less. The prediction also stays close to what was actually observed. At $K = 2$ it gives back the second temporary update exactly, and for larger K it is exact whenever later motions follow the trend. When they do not, Section 6 bounds the error. Algorithm 1 lists these steps, Theorem 2 and Appendix A.5 derive the endpoint, and Appendix A.3 the safeguards on R.

5.3 Reposition and Correct

The predicted endpoint is an estimate, so PrePO does not jump to it directly. It moves a fraction α of the way there from the original parameters, $\tilde{\theta} = \theta_0 + \alpha(\hat{\theta} - \theta_0)$. Reaching this point costs nothing extra, because it only combines parameters that PrePO already holds. At the new point PrePO computes a fresh gradient of the same batch loss and takes one real optimizer step from there, starting from the saved optimizer state s_0. The prediction only decides where the loss is evaluated, and the update itself always comes from a real gradient. The two temporary updates are then discarded, and only the corrected parameters and their optimizer state carry into the next iteration. In total, one PrePO step uses three backward passes, two for the temporary updates and one for the correction, however large K is.

6 Prediction Fidelity and Convergence Guarantees

PrePO rests on a prediction, so two questions follow: how close is the predicted endpoint to where K real updates would land, and does training still converge when the prediction is imperfect? For the first, write each motion as the trend applied to the previous one plus a leftover, $z_{j+1}^t = Rz_j^t + e_j^t$. This is an identity rather than an assumption, so the endpoint error is an explicit sum of leftovers.

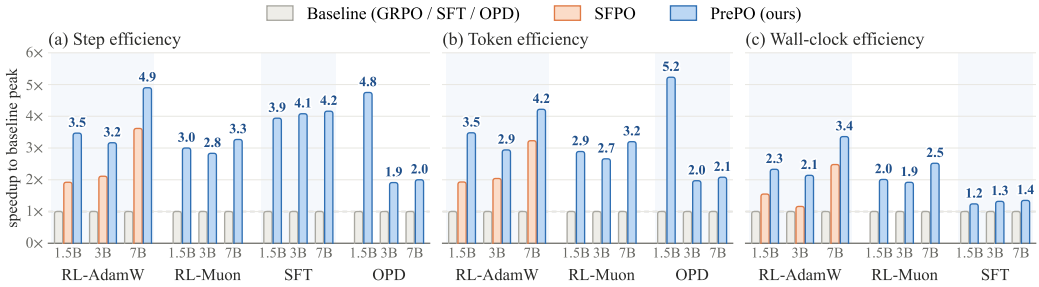

Figure 2: Speedup in reaching the baseline's peak for every setting in Tables 2 and 3; higher is better and the baseline is 1×. RL uses Qwen2.5-Math at 1.5B and 7B and Llama-3.2 at 3B, SFT uses Llama-3.2 at 3B and Qwen2.5 otherwise, and OPD uses Qwen2.5. SFT tokens were not recorded, and OPD wall-clock is omitted.

When $\|R\| \le 1$, $\|C\| \le 1$, and every leftover is at most ε_e, Theorem 1 gives

$$\left\|\theta^{t,K} - \widehat{\theta}^{t,K}\right\| \;\le\; \frac{(K-1)(K-2)}{2}\,\|A\|\,\varepsilon_e, \tag{4}$$

where $\|A\| = 1$ for Muon and $\|A\| = \eta$ for AdamW, and Lemma 2 computes ε_e for AdamW from quantities measured during training. The same identity shows that prediction is not a larger step. Each coordinate gets its own step size, the full $\gamma_{\text{hi}} = \alpha K\eta + \eta_c$ where its motion persists and at most $2\alpha\eta + \eta_c$ where it reverses, which keeps PrePO stable at curvatures where a uniformly larger step diverges, as Proposition 2 shows. Under a Polyak–Łojasiewicz condition, one PrePO step on a batch therefore makes the progress of 1.5 to 3.5 plain steps in our setting (Proposition 3). For the second question, Theorems 3 and 4 prove convergence to a neighbourhood of a stationary point, set by gradient bias and clipping, for AdamW and Muon with fresh on-policy batches, the clipped surrogate, gradient clipping, the gate, and optimizer state carried across batches. They assume a smooth target (Zhang et al., 2020; Agarwal et al., 2021), bounded gradient bias and clipping drift, and, for AdamW, a preconditioner held fixed within each outer step. By Corollary 1, the AdamW rate is the usual $O(1/\sqrt{T})$ plus two costs of prediction, a variance term from fitting the trend on the same batch and a drift term bounded by the clip fraction, the first vanishing at $\alpha = 0$ and the second whenever no token is clipped. For Muon the rate is $O(T^{-1/4})$ in the nuclear norm. Appendices A and B give the full statements and proofs, and Table 6 compares measured endpoint errors with the bound.

7 Experiments

7.1 Experimental Protocol

We test PrePO where reusing a batch matters most: reinforcement learning with verifiable rewards on math reasoning. The training pool has 25,417 prompts, 17,917 from DAPO-Math-17K (Yu et al., 2025) and 7,500 from the training split of MATH (Hendrycks et al., 2021). The main comparison uses Qwen2.5-Math-1.5B-Instruct and Qwen2.5-Math-7B-Instruct (Yang et al., 2024) together with Llama-3.2-3B-Instruct (Meta AI, 2024), so it covers two model families and three sizes. To keep the comparison fair, GRPO, SFPO, and PrePO share the same prompt stream, learning rate, rollout budget, reward function, and hardware, and every main RL run trains for 400 steps with 256 prompts per batch and eight responses per prompt. PrePO uses $K = 5$ with $\alpha = 0.5$ for RL, with AdamW or Muon, and $\alpha = 0.3$ for SFT and OPD, all fixed before any evaluation. Responses are capped at 3,072 generated tokens in training and evaluation. We evaluate every checkpoint with greedy Pass@1 and report each method at its final checkpoint on MATH-500 (Hendrycks et al., 2021; Lightman et al., 2024), AIME24 (Art of Problem Solving, 2024a;b), AIME25 (Art of Problem Solving, 2025a;b), AMC23 (Art of Problem Solving, n.d.), MinervaMATH (Lewkowycz et al., 2022), and OlympiadBench (He et al., 2024). To see whether the benefit depends on the RL loss or on AdamW, we then change one ingredient at a time. We replace AdamW with Muon (Jordan, 2024) on Qwen2.5-

Math-1.5B-Instruct, Llama-3.2-3B-Instruct, and Qwen2.5-Math-7B-Instruct (Yang et al., 2024), we switch to supervised fine-tuning on the knights-and-knaves logic puzzles (Xie et al., 2025), a dataset separate from the math pool, and we switch to on-policy distillation (Agarwal et al., 2024) with a Qwen2.5-Math-7B-Instruct teacher for Qwen2.5-1.5B and Qwen2.5-3B, and a Qwen2.5-32B-Instruct teacher for Qwen2.5-7B, showing how the gain carries across model scale. Appendix C.1 lists every remaining setting.

Table 2: RLVR efficiency and accuracy. Shaded columns give the cost of reaching GRPO's peak score. Scores use greedy evaluation (Pass@1), with responses capped at 3,072 tokens; Avg. is the mean over the six benchmarks.

Method	Efficiency			Accuracy						
	Steps to GRPO Peak ↑	Token Speedup ↑	Wall-clock Speedup ↑	Avg. Pass@1 ↑	Math -500	AIME 24	AIME 25	AMC 23	Minerva Math	Olympiad Bench
Qwen2.5-Math-1.5B										
GRPO	392	1.00×	1.00×	36.29	74.00	13.33	10.00	53.01	27.94	39.47
SFPO	$204^{+1.92\times}$	1.93×	1.55×	$38.50^{+2.21}$	$76.40^{+2.40}$	$16.67^{+3.34}$	$13.33^{+3.33}$	$55.42^{+2.41}$	$29.41^{+1.47}$	$39.76^{+0.29}$
PrePO	$\mathbf{113}^{+3.47\times}$	**3.48**×	**2.33**×	$\mathbf{40.30}^{+4.01}$	$\mathbf{77.80}^{+3.80}$	$\mathbf{20.00}^{+6.67}$	$\mathbf{16.67}^{+6.67}$	$\mathbf{56.63}^{+3.62}$	$\mathbf{30.51}^{+2.57}$	$\mathbf{40.21}^{+0.74}$
Llama-3.2-3B-Instruct										
GRPO	285	1.00×	1.00×	22.30	52.80	6.67	0.00	36.14	20.22	17.95
SFPO	$135^{+2.11\times}$	2.04×	1.16×	$24.09^{+1.79}$	$55.20^{+2.40}$	$10.00^{+3.33}$	0.00	$38.55^{+2.41}$	$21.32^{+1.10}$	$19.44^{+1.49}$
PrePO	$\mathbf{90}^{+3.17\times}$	**2.94**×	**2.14**×	$\mathbf{26.25}^{+3.95}$	$\mathbf{57.20}^{+4.40}$	$\mathbf{13.33}^{+6.66}$	$\mathbf{3.33}^{+3.33}$	$\mathbf{40.96}^{+4.82}$	$\mathbf{22.06}^{+1.84}$	$\mathbf{20.62}^{+2.67}$
Qwen2.5-Math-7B-Instruct										
GRPO	387	1.00×	1.00×	45.83	85.40	20.00	16.67	72.29	35.66	44.96
SFPO	$107^{+3.62\times}$	3.23×	2.48×	$46.91^{+1.08}$	$85.60^{+0.20}$	$23.33^{+3.33}$	16.67	$\mathbf{74.70}^{+2.41}$	$36.03^{+0.37}$	$45.10^{+0.14}$
PrePO	$\mathbf{79}^{+4.90\times}$	**4.22**×	**3.36**×	$\mathbf{49.17}^{+3.34}$	$\mathbf{87.60}^{+2.20}$	$\mathbf{26.67}^{+6.67}$	$\mathbf{20.00}^{+3.33}$	$\mathbf{74.70}^{+2.41}$	$\mathbf{38.60}^{+2.94}$	$\mathbf{47.48}^{+2.52}$

7.2 Fewer Steps, Fewer Rollouts, Less Time

If one batch moves the model as far as several updates, we need fewer batches, and each batch is one round of rollouts. Table 2 checks each part against the peak score of GRPO. We count steps as the number of rollout batches used. PrePO needs $3.2\times$ to $4.9\times$ fewer steps across the three models. The number of generated tokens falls by almost the same amount (Table 5). We measure wall-clock time with rollouts included on four H100 GPUs. It falls by $2.1\times$ to $3.4\times$. The time gain is smaller than the step gain because a predicting step runs three backward passes. Yet a PrePO step takes only about 1.5 times as long as a GRPO step on every model. It is not three times as long because rollout generation takes most of each step, and PrePO never repeats it. PrePO keeps its saved parameters and optimizer state in CPU memory, and these times include moving them. SFPO is the baseline that moves farther per batch by running all K updates for real, and it needs more steps, tokens, and time than PrePO. If $3.5\times$ fewer batches reach the same score, one update uses only part of what a batch can teach. So the limit is how well we use each batch and not how much data we collect. Figure 2 shows that PrePO needs fewer steps and tokens than every baseline and than SFPO in every setting.

7.3 Why PrePO Also Finishes Higher

PrePO was built to save cost. Yet it also ends with the best average score on every model. All methods share the same fixed budget of 400 steps. In Figure 3, GRPO flattens over the second half and peaks late, at step 392 on Qwen2.5-Math-1.5B and 285 on Llama-3.2-3B-Instruct. PrePO reaches that level in a third of the batches or fewer and keeps improving with the steps it saves. Each PrePO update makes the progress of up to 3.5 plain updates (Section 6). Near a flat peak a small gain in progress saves many steps, so the step saving can exceed this, as the $4.9\times$ on Qwen2.5-Math-7B shows. Each coordinate gets its own step size, the full K steps where its motion persists and a shorter one where the probes show overshooting (Proposition 2). Figure 4 shows that most motion

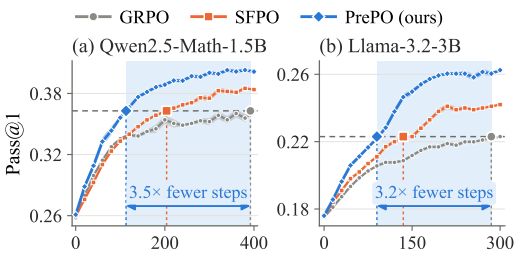

Figure 3: Time to reach GRPO's peak, dashed, on Qwen2.5-Math-1.5B, panel a, and Llama-3.2-3B-Instruct, panel b. Blue shading marks the steps PrePO saves.

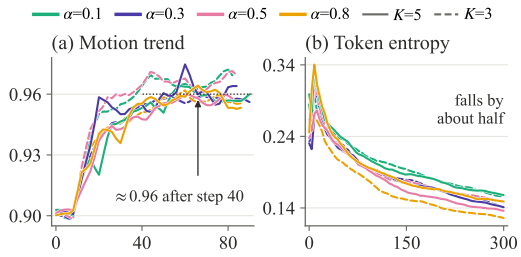

Figure 4: Muon runs over training steps: the fitted motion trend, panel a, and token entropy, panel b. Color marks α and line style marks K.

Table 3: Transfer efficiency and accuracy across optimizers and objectives. Shaded columns give the cost of reaching the baseline's target (in SFT, reaching the baseline's step-400 score under a compute-constrained budget; n is the number of people in a puzzle); OPD wall-clock is approximate. Scores use greedy evaluation (Pass@1; RL responses capped at 3,072 tokens); Avg. and Overall are the mean over the six test columns.

		Efficiency			Accuracy						
Reinforcement Learning (Muon optimizer)											
Model	Method	Steps to GRPO Peak ↑	Token Speedup ↑	Wall-clock Speedup ↑	Avg. Pass@1 ↑	Math -500	AIME 24	AIME 25	AMC 23	Minerva Math	Olympiad Bench
Qwen-Math 1.5B	GRPO	345	1.00×	1.00×	37.62	75.80	10.00	13.33	55.42	29.78	41.39
	PrePO	**115**$^{+3.00\times}$	**2.89**×	**2.01**×	**40.34**$^{+2.72}$	**78.80**$^{+3.00}$	**16.67**$^{+6.67}$	**16.67**$^{+3.34}$	**56.63**$^{+1.21}$	**30.51**$^{+0.73}$	**42.73**$^{+1.34}$
Llama 3B	GRPO	255	1.00×	1.00×	22.85	50.20	**16.67**	**10.00**	26.51	18.01	15.73
	PrePO	**90**$^{+2.83\times}$	**2.66**×	**1.92**×	**26.44**$^{+3.59}$	**51.60**$^{+1.40}$	13.33$^{-3.34}$	**10.00**	**40.96**$^{+14.45}$	**20.96**$^{+2.95}$	**21.81**$^{+6.08}$
Qwen-Math 7B	GRPO	340	1.00×	1.00×	47.57	86.60	23.33	16.67	74.70	37.50	46.59
	PrePO	**104**$^{+3.27\times}$	**3.20**×	**2.52**×	**49.50**$^{+1.93}$	**88.00**$^{+1.40}$	**26.67**$^{+3.34}$	**20.00**$^{+3.33}$	**75.90**$^{+1.20}$	**38.97**$^{+1.47}$	**47.48**$^{+0.89}$
Supervised Fine-Tuning											
Model	Method	Step Gain ↑	Token Speedup ↑	Time Gain ↑	Avg. Pass@1 ↑	IID n=3	IID n=4	IID n=5	IID n=6	OOD n=7	OOD n=8
Qwen 1.5B	SFT	1.00×	–	1.00×	7.33	22.00	12.00	5.00	3.00	1.00	1.00
	PrePO	**3.94**×	–	**1.24**×	**18.33**$^{+11.00}$	**40.00**$^{+18.00}$	**27.00**$^{+15.00}$	**19.00**$^{+14.00}$	**12.00**$^{+9.00}$	**5.00**$^{+4.00}$	**7.00**$^{+6.00}$
Llama 3B	SFT	1.00×	–	1.00×	15.33	35.00	25.00	17.00	6.00	5.00	4.00
	PrePO	**4.08**×	–	**1.32**×	**33.00**$^{+17.67}$	**60.00**$^{+25.00}$	**46.00**$^{+21.00}$	**36.00**$^{+19.00}$	**29.00**$^{+23.00}$	**15.00**$^{+10.00}$	**12.00**$^{+8.00}$
Qwen 7B	SFT	1.00×	–	1.00×	23.50	50.00	38.00	26.00	14.00	8.00	5.00
	PrePO	**4.16**×	–	**1.35**×	**43.33**$^{+19.83}$	**74.00**$^{+24.00}$	**60.00**$^{+22.00}$	**48.00**$^{+22.00}$	**38.00**$^{+24.00}$	**23.00**$^{+15.00}$	**17.00**$^{+12.00}$
On-Policy Distillation (OPD)											
Model	Method	Steps to OPD Peak ↑	Token Speedup ↑	Wall-clock Speedup ↑	Overall	Math -500	AIME 24	AIME 25	AMC 23	Minerva Math	Olympiad Bench
Qwen 1.5B	OPD	95	1.00×	1.00×	21.56	54.40	3.33	3.33	27.71	19.49	21.07
	PrePO	**20**$^{+4.75\times}$	**5.23**×	**1.58**×	**24.00**$^{+2.44}$	**58.40**$^{+4.00}$	**6.67**$^{+3.34}$	**6.67**$^{+3.34}$	**30.12**$^{+2.41}$	**19.85**$^{+0.36}$	**22.26**$^{+1.19}$
Qwen 3B	OPD	65	1.00×	**1.00**×	26.83	62.80	6.67	3.33	33.73	24.63	29.82
	PrePO	**34**$^{+1.91\times}$	**1.97**×	0.79×	**29.32**$^{+2.49}$	**64.60**$^{+1.80}$	**10.00**$^{+3.33}$	**6.67**$^{+3.34}$	**36.14**$^{+2.41}$	**27.94**$^{+3.31}$	**30.56**$^{+0.74}$
Qwen 7B	OPD	52	1.00×	**1.00**×	41.27	78.60	20.00	13.33	59.04	34.19	42.43
	PrePO	**26**$^{+2.00\times}$	**2.08**×	0.82×	**43.10**$^{+1.83}$	**80.20**$^{+1.60}$	**23.33**$^{+3.33}$	**16.67**$^{+3.34}$	**60.24**$^{+1.20}$	**35.29**$^{+1.10}$	**42.88**$^{+0.45}$

keeps its direction. The fitted Muon trend stays near 0.96 and almost no matrix has a negative trend. The best α depends on K. It is 0.5 at $K = 3$, and Muon prefers 0.3 at $K = 5$, but the main runs keep the pre-fixed $\alpha = 0.5$ (Appendix C.2). Token entropy falls as accuracy rises (Cui et al., 2025; Wang et al., 2026b), and a sharp change in it sets α to zero for the rest of training. Removing this fallback drops the mean accuracy from 38.9 to 31.1, and a temporary one reaches 37.7 (Appendix C.6).

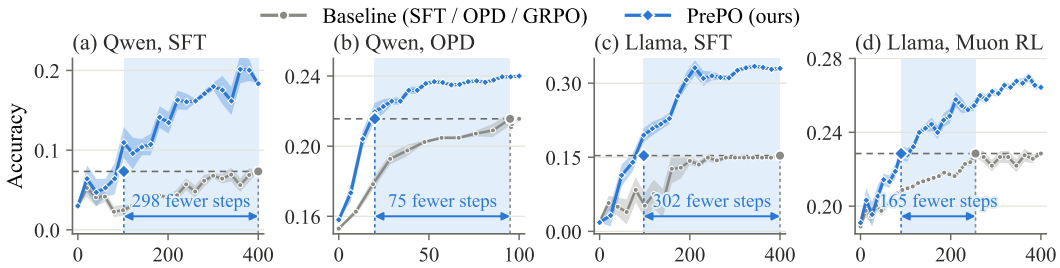

Figure 5: Time to each baseline's target, dashed (its peak; for SFT its step-400 score): SFT and OPD on Qwen2.5-1.5B, a and b, and SFT and Muon RL on Llama-3.2-3B, c and d. Shading marks the steps PrePO saves.

7.4 Transfer Beyond RLVR and Across Optimizers

The probes only read how the optimizer moves, so the gain should not depend on the RL loss or on AdamW. We swap AdamW for Muon and also run SFT and on-policy distillation, each against its own baseline. A code-generation run (Appendix C.7) extends RLVR beyond math. In Figure 5 and Table 3, PrePO reaches every target $1.9\times$ to $4.8\times$ sooner in steps. It also ends higher in every case, most clearly in SFT. With the same data, learning rate of 10^{-5}, and 400 steps, the SFT baseline reaches only 7.3, 15.3, and 23.5 average accuracy. PrePO reaches 18.3, 33.0, and 43.3. PrePO reaches the baseline's step-400 score in only 96 to 102 steps. The gain also holds on test puzzles larger than any training puzzle. In SFT each step is one pass over a batch, so the $4\times$ step saving reaches the same score from a quarter of the examples. Without a costly rollout round a PrePO step costs about three plain steps, so the $4\times$ step saving becomes $1.2\times$ to $1.4\times$ in time (Table 3). OPD samples one response per prompt, so a PrePO step there costs 2.4 to 3 plain steps, more than its $2\times$ saving on 3B and 7B.

7.5 Prediction Holds Across Model Size

Looking K steps ahead does not cost K steps of compute. SFPO pays $K+1$ backward passes per update. PrePO pays three for any K. Our logs confirm this at $K = 5$ and $K = 10$. Nor is the path too irregular to predict. We run the real K steps next to the prediction on the same batch. The prediction points the same way as the real move, with a cosine above 0.9 even at $K = 10$. The prediction also lands two to five times closer than stopping after the two probes. We then move from Qwen2.5-Math-1.5B to Qwen2.5-3B. The real move becomes 1.4 to 1.7 times longer. Yet the error of PrePO stays the same share of that move (Figure 6). So the error depends on the horizon and not on model size, as the bound predicts (Appendix C.5).

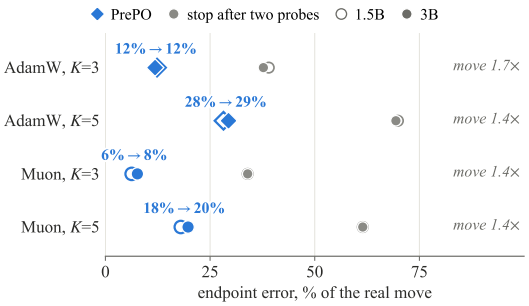

Figure 6: Median endpoint error as a share of the real K-step move, 1.5B hollow and 3B filled.

8 Discussion and Conclusion

Over a few updates the optimizer path is regular enough to predict from two probes. What sets it apart is the cost. SFPO runs every inner step, so its cost grows with K. PrePO looks K steps ahead for three backward passes and lets a real gradient make the update. Our theory shows that prediction keeps the usual convergence rate and that one PrePO step makes up to 3.5 plain steps of progress, and our experiments confirm the speedup. PrePO needs fewer steps and ends higher in every setting. It helps most where batches are costly or budgets are tight.

AI USE STATEMENT

We used generative AI tools to obtain feedback on experimental design and to assist with the analysis of experimental findings. These tools also supported literature searches, analysis code, figure preparation, and the drafting and editing of parts of the manuscript. The authors reviewed all AI-assisted material and take responsibility for the methods, results, claims, code, and figures in this paper.

ETHICS STATEMENT

PrePO lowers the cost of post-training without changing the data, reward, loss, or optimizer. In our main RL runs it needs a third of the rollout batches or fewer, which saves GPU hours and energy and brings RL post-training within reach of smaller budgets. Cheaper training also makes misuse cheaper, so the usual safeguards for releasing models still apply.

REPRODUCIBILITY STATEMENT

All tables and figures in this paper are reproducible from the settings in Appendix C.1, and we will release the code, scripts, and random seeds used to produce them. The assumptions and complete proofs of all theoretical results are in Appendices A and B.

REFERENCES

Alekh Agarwal, Sham M. Kakade, Jason D. Lee, and Gaurav Mahajan. On the theory of policy gradient methods: Optimality, approximation, and distribution shift. *Journal of Machine Learning Research*, 22(98):1–76, 2021.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*, 2024.

Alexander C. Aitken. On Bernoulli's numerical solution of algebraic equations. *Proceedings of the Royal Society of Edinburgh*, 46:289–305, 1927. doi: 10.1017/S0370164600022070.

Donald G. Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM*, 12 (4):547–560, 1965.

Art of Problem Solving. 2024 AIME I problems and solutions. `https://artofproblemsolving.com/wiki/index.php/2024_AIME_I`, 2024a.

Art of Problem Solving. 2024 AIME II problems and solutions. `https://artofproblemsolving.com/wiki/index.php/2024_AIME_II`, 2024b.

Art of Problem Solving. 2025 AIME I problems and solutions. `https://artofproblemsolving.com/wiki/index.php/2025_AIME_I`, 2025a.

Art of Problem Solving. 2025 AIME II problems and solutions. `https://artofproblemsolving.com/wiki/index.php/2025_AIME_II`, 2025b.

Art of Problem Solving. Amc problems and solutions. `https://artofproblemsolving.com/wiki/index.php?title=AMC`, n.d. Accessed 2025-04-20.

Jeremy Bernstein and Laker Newhouse. Old optimizer, new norm: An anthology, 2024.

Stanley Cabay and L. W. Jackson. A polynomial extrapolation method for finding limits and antilimits of vector sequences. *SIAM Journal on Numerical Analysis*, 13(5):734–752, 1976.

Yuchen Cai, Ding Cao, Xin Xu, Zijun Yao, Yuqing Huang, Benyi Zhang, Zhenyu Tan, Guiquan Liu, and Junfeng Fang. On predictability of reinforcement learning dynamics for large language models. In *International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=SdHmA6BYVJ`.

Chi-Chung Chen, Chia-Lin Yang, and Hsiang-Yun Cheng. Efficient and robust parallel DNN training through model parallelism on multi-GPU platform, 2018.

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.

Xiangyi Chen, Sijia Liu, Ruoyu Sun, and Mingyi Hong. On the convergence of a class of Adam-type algorithms for non-convex optimization. In *ICLR*, 2019.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Ashok Cutkosky and Harsh Mehta. Momentum improves normalized SGD. In *ICML*, 2020.

Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. Training GANs with optimism. In *ICLR*, 2018.

Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. A simple convergence proof of Adam and Adagrad. *Transactions on Machine Learning Research*, 2022.

Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent, and Simon Lacoste-Julien. A variational inequality perspective on generative adversarial networks. In *ICLR*, 2019.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *ICLR*, 2024.

Lei Guan, Dongsheng Li, Yanqi Shi, and Jian Meng. XGrad: Boosting gradient-based optimizers with weight prediction, 2023.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Leng Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3828–3850, Bangkok, Thailand, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

Jingkai He, Tianjian Li, Erhu Feng, Dong Du, Qian Liu, Tao Liu, Yubin Xia, and Haibo Chen. History rhymes: Accelerating LLM reinforcement learning with RhymeRL. *arXiv preprint arXiv:2508.18588*, 2025.

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 *NeurIPS*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, et al. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Heyang Jiang, Henry Liu, and Baharan Mirzasoleiman. Learning as reasoning unfolds: Progressive rollout allocation for efficient reinforcement learning, 2026. URL https://arxiv.org/abs/2607.22002.

Keller Jordan. Muon: An optimizer for hidden layers in neural networks. https://kellerjordan.github.io/posts/muon/, 2024.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the Polyak–Łojasiewicz condition. In *Machine Learning and Knowledge Discovery in Databases*, pp. 795–811. Springer International Publishing, 2016. doi: 10.1007/978-3-319-46128-1_50. URL https://arxiv.org/abs/1608.04636.

Minseo Kim, Minjae Lee, Seunghyuk Oh, Kevin Galim, Donghoon Kim, Coleman Hooper, Harman Singh, Amir Gholami, Hyung Il Koo, and Wonjun Kang. EfficientRollout: System-aware self-speculative decoding for RL rollouts. *arXiv preprint arXiv:2606.18967*, 2026.

G. M. Korpelevich. The extragradient method for finding saddle points and other problems. *Ekonomika i Matematicheskie Metody*, 12:747–756, 1976.

Guanghui Lan and Yi Zhou. Random gradient extrapolation for distributed and stochastic optimization. *SIAM Journal on Optimization*, 28(4):2753–2782, 2018.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. In *NeurIPS*, 2022.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024.

Tao Lin, Lingjing Kong, Sebastian U. Stich, and Martin Jaggi. Extrapolation for large-batch training in deep learning. In *ICML*, 2020.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation. In *NeurIPS*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Jerry Ma and Denis Yarats. Quasi-hyperbolic momentum and Adam for deep learning. In *ICLR*, 2019.

Luckeciano C Melo, Alessandro Abate, and Yarin Gal. Stabilizing policy gradients for sample-efficient reinforcement learning in LLM reasoning. In *ICLR*, 2026.

Panayotis Mertikopoulos, Bruno Lecouat, Houssam Zenati, Chuan-Sheng Foo, Vijay Chandrasekhar, and Georgios Piliouras. Optimistic mirror descent in saddle-point problems: Going the extra (gradient) mile. In *ICLR*, 2019.

Meta AI. Llama 3.2 model card. https://github.com/meta-llama/llama-models/blob/main/models/llama3_2/MODEL_CARD.md, 2024.

Aryan Mokhtari, Asuman Ozdaglar, and Sarath Pattathil. A unified analysis of extra-gradient and optimistic gradient methods for saddle point problems: Proximal point approach. In *AISTATS*, 2020.

Youssef Mroueh. Reinforcement learning with verifiable rewards: GRPO's effective loss, dynamics, and success amplification. *arXiv preprint arXiv:2503.06639*, 2025.

Youssef Mroueh, Nicolas Dupuis, Brian Belgodere, Apoorva Nitsure, Mattia Rigotti, Kristjan Greenewald, Jiri Navratil, Jerret Ross, and Jesus Rios. Revisiting group relative policy optimization: Insights into on-policy and off-policy training. *arXiv preprint arXiv:2505.22257*, 2025.

Arkadi Nemirovski. Prox-method with rate of convergence $O(1/t)$ for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. *SIAM Journal on Optimization*, 15(1):229–251, 2004.

Yurii Nesterov. A method for solving the convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376, 1983.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35, 2022.

Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Leonid Denisovich Popov. A modification of the Arrow–Hurwicz method for search of saddle points. *Mathematical Notes of the Academy of Sciences of the USSR*, 28(5):845–848, 1980.

Kimon Protopapas and Anas Barakat. Policy mirror descent with lookahead. In *Advances in Neural Information Processing Systems*, volume 37, pp. 26443–26481. Curran Associates, Inc., 2024. doi: 10.52202/079017-0833. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/2e8eaf43f20948ad878e6b8902797d1e-Paper-Conference.pdf.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report, 2024. URL https://arxiv.org/abs/2412.15115.

Alexander Rakhlin and Karthik Sridharan. Optimization, learning, and games with predictable sequences. In *NeurIPS*, 2013.

Lewis Fry Richardson. The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam. *Philosophical Transactions of the Royal Society of London, Series A*, 210:307–357, 1911.

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

Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. In *NeurIPS*, 2016.

Zhihong Shao, Peiyi Wang, Qihao Zhu, et al. 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 '25)*, 2025.

Avram Sidi, William F. Ford, and David A. Smith. Acceleration of convergence of vector sequences. *SIAM Journal on Numerical Analysis*, 23(1):178–196, 1986.

Homer F. Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

Tianle Wang, Jiayu Liu, Zhongyuan Wu, Shenghao Jin, Wei Chen, Hao Xu, and Ning Miao. Linear dynamics in the RLVR training of large language models. *arXiv preprint arXiv:2601.04537*, 2026a.

Ziyan Wang, Zheng Wang, Xingwei Qu, Qi Cheng, Jie Fu, Shengpu Tang, Minjia Zhang, and Xiaoming Huo. Slow-fast policy optimization: Reposition-before-update for LLM reasoning. In *International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=xBlHiHdXap.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners. In *ICLR*, 2022.

Jiahao Wu, Ning Lu, Shengcai Liu, Kun Wang, Yanting Yang, Baijiong Lin, Chen Jason Zhang, Qing Li, and Ke Tang. Train at moving edge: Online-verified prompt selection for efficient RL training of large reasoning model, 2026. URL https://arxiv.org/abs/2603.25184.

Chulin Xie, Yangsibo Huang, Chiyuan Zhang, Da Yu, Xinyun Chen, Bill Yuchen Lin, Bo Li, Badih Ghazi, and Ravi Kumar. On memorization of large language models in logical reasoning. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 2742–2785, Mumbai, India, dec 2025. The Asian Federation of Natural Language Processing and The Association for Computational Linguistics. doi: 10.18653/v1/2025.ijcnlp-long.148. URL https://aclanthology.org/2025.ijcnlp-long.148/.

Shuo Xie and Zhiyuan Li. Implicit bias of AdamW: ℓ_∞ norm constrained optimization. In *International Conference on Machine Learning*, 2024.

Yixuan Even Xu, Yash Savani, Fei Fang, and J. Zico Kolter. Not all rollouts are useful: Down-sampling rollouts in LLM reinforcement learning. *Transactions on Machine Learning Research*, 2026.

Zhangchen Xu, Yuetai Liu, Yao Yin, Ming Zhou, and Radha Poovendran. KodCode: A diverse, challenging, and verifiable synthetic dataset for coding. In *Findings of ACL*, 2025.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement, 2024. URL https://arxiv.org/abs/2409.12122.

Zihao Ye, Lequn Chen, Ruihang Lai, Wuwei Lin, Yineng Zhang, Stephanie Wang, Tianqi Chen, Baris Kasikci, Vinod Grover, Arvind Krishnamurthy, and Luis Ceze. Flashinfer: Efficient and customizable attention engine for LLM inference serving. In *Proceedings of Machine Learning and Systems*, volume 7, 2025. URL https://proceedings.mlsys.org/paper_files/paper/2025/file/dbf02b21d77409a2db30e56866a8ab3a-Paper-Conference.pdf.

Qiying Yu, Zheng Zhang, Ruofei Zhu, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *NeurIPS*, 2025.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, Xiangpeng Wei, Xiangyu Yu, Gaohong Liu, Juncai Liu, Lingjun Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Ru Zhang, Xin Liu, Mingxuan Wang, Yonghui Wu, and Lin Yan. VAPO: Efficient and reliable reinforcement learning for advanced reasoning tasks, 2025. URL https://arxiv.org/abs/2504.05118.

Kaiqing Zhang, Alec Koppel, Hao Zhu, and Tamer Başar. Global convergence of policy gradient methods to (almost) locally optimal policies. *SIAM Journal on Control and Optimization*, 58(6): 3586–3612, 2020.

Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. In *NeurIPS*, 2019.

Chujie Zheng, Ziqi Wang, Heng Ji, Minlie Huang, and Nanyun Peng. Weak-to-strong extrapolation expedites alignment. *arXiv preprint arXiv:2404.16792*, 2024.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. Group sequence policy optimization, 2025a. URL https://arxiv.org/abs/2507.18071.

Haizhong Zheng, Yang Zhou, Brian R. Bartoldson, Bhavya Kailkhura, Fan Lai, Jiawei Zhao, and Beidi Chen. Act only when it pays: Efficient reinforcement learning for LLM reasoning via selective rollouts. In *Advances in Neural Information Processing Systems*, 2025b. doi: 10.52202/085713-4143. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/b42fb82fc3ffb88b872c4714f093875d-Abstract-Conference.html.

Zhenxun Zhuang, Mingrui Liu, Ashok Cutkosky, and Francesco Orabona. Understanding AdamW through proximal methods and scale-freeness. *Transactions on Machine Learning Research*, 2022.

A Endpoint Prediction: Identity, Bound, and Optimizer Forms

This appendix writes PrePO's endpoint error exactly, bounds it, and gives the AdamW and Muon forms of the predictor. Appendix B then bounds the residuals and proves convergence under RLVR training.

A.1 Reference trajectory and notation

Fix $K \geq 2$, a starting point (θ_0, s_0), a batch, and all randomness. The reference trajectory applies K real transitions $(\theta_{n+1}, s_{n+1}) = \Phi(\theta_n, s_n, \bar{g}_n)$ on this batch; the first two are the probes, so θ_1, θ_2 and the motions z_0, z_1 are observed. PrePO discards the probe states and restores s_0 before the correction, so the probes reach the committed update only through the repositioned point $\widetilde{\theta}$. Norms are Euclidean for vectors, spectral for operators, and Frobenius for matrix motions.

Assumption 1 (Fixed transition representation)**.** *Fixed matrices $C, A \in \mathbb{R}^{d\times d}$ and motions $z_n \in \mathbb{R}^d$ represent the transitions as $\theta_{n+1} = C\theta_n + Az_n$ for $0 \leq n < K$.*

Fit a trend R from the two observed motions and hold it fixed. The residual and the propagation operator are

$$e_n := z_{n+1} - Rz_n, \qquad M_m := \sum_{q=0}^{m-1} C^{m-1-q} A R^q, \qquad M_0 = 0. \tag{5}$$

So $z_{n+1} = Rz_n + e_n$ holds by definition, not by assumption. For displacement motions ($C = A = I$), $M_m = S_m(R) = \sum_{q<m} R^q$, which for a scalar $r \neq 1$ is $(1 - r^m)/(1 - r)$ and equals m at $r = 1$.

A.2 Endpoint identity and bound

Theorem 1 (Anchored endpoint decomposition)**.** *Under Assumption 1, for $0 \leq j < K$ let $\widehat{\theta}_K^{(j)} = C^{K-j}\theta_j + M_{K-j} z_j$. For any fixed R,*

$$\theta_K - \widehat{\theta}_K^{(j)} = \sum_{\ell=j}^{K-2} M_{K-1-\ell} e_\ell, \qquad \|\theta_K - \widehat{\theta}_K^{(j)}\| \leq \sum_{\ell=j}^{K-2} \|M_{K-1-\ell}\| \|e_\ell\|. \tag{6}$$

Proof. Unrolling both recurrences from step j gives

$$z_n = R^{n-j} z_j + \sum_{\ell=j}^{n-1} R^{n-1-\ell} e_\ell, \qquad \theta_K = C^{K-j}\theta_j + \sum_{n=j}^{K-1} C^{K-1-n} A z_n. \tag{7}$$

Substitute the first into the second. The z_j terms sum to $M_{K-j} z_j$, and after swapping the double sum each e_ℓ collects the coefficient $\sum_{q=0}^{K-2-\ell} C^{K-2-\ell-q} A R^q = M_{K-1-\ell}$. The bound is the triangle inequality. □

PrePO anchors at the second motion, $j = 1$:

$$\widehat{\theta}_K = C^{K-1}\theta_1 + M_{K-1} z_1, \qquad \theta_K - \widehat{\theta}_K = \sum_{\ell=1}^{K-2} M_{K-1-\ell} e_\ell. \tag{8}$$

This anchor needs no inverse and carries no e_0 term. It is exact at $K = 2$, and at $K = 3$ the whole error is Ae_1.

For scalars write $T_m(a,b) = \sum_{q<m} a^{m-1-q}b^q$, which equals $(a^m - b^m)/(a-b)$, or ma^{m-1} when $a = b$. With $c_* = \|C\|$ and $r_* = \|R\|$, submultiplicativity gives $\|M_m\| \le \|A\| T_m(c_*, r_*)$, so

$$\|\theta_K - \widehat{\theta}_K\| \le \|A\| \sum_{\ell=1}^{K-2} T_{K-1-\ell}(c_*, r_*)\|e_\ell\|. \tag{9}$$

When $c_*, r_* \le 1$ each term of T_m is at most one, and if every $\|e_\ell\| \le \varepsilon_e$,

$$\|\theta_K - \widehat{\theta}_K\| \le \tfrac{1}{2}(K-1)(K-2)\|A\|\varepsilon_e. \tag{10}$$

Clipping a diagonal or scalar trend to $[-1, 1]$ gives $\|R\| \le 1$. The residual bound ε_e is not left as an assumption; Lemma 2 computes it from optimizer constants and quantities logged during training. Some such information is needed, because the probes alone cannot bound the error:

Proposition 1 (Two-probe identifiability limit)**.** *For unrestricted continuations and $K \ge 3$, no finite error bound depends only on θ_0, z_0, z_1, even if $e_0 = 0$.*

Proof. Take scalar $C = A = R = 1$, $z_0 = z_1 = 1$, and $z_n = 1 + v$ for $n \ge 2$. The probes do not see v, yet the error is $(K-2)v$. □

A.3 Retention fit

One pair (z_0, z_1) cannot identify a full matrix R, so PrePO fits a diagonal trend for AdamW and a scalar trend for Muon. The diagonal fit takes the coordinatewise ratio, projected onto $[r_{\min}, r_{\max}]$, and falls back to r_{neu} where the first motion is too small to divide by:

$$r_i = \begin{cases} \Pi_{[r_{\min}, r_{\max}]}(z_{1,i}/z_{0,i}), & |z_{0,i}| > \delta, \\ r_{\text{neu}}, & \text{otherwise}, \end{cases} \qquad R = \operatorname{diag}(r_i). \tag{11}$$

The projected ratio is the constrained least-squares fit of $z_{1,i} \approx r z_{0,i}$. The scalar fit is $\rho = \langle z_0, z_1\rangle / \|z_0\|^2$, projected the same way and set to 0 when $z_0 = 0$. We use $[r_{\min}, r_{\max}] = [-1, 1]$, $r_{\text{neu}} = 1$, and $\delta = 10^{-3}\,\mathrm{RMS}(z_0)$. The interval contains the multiplier $1 - \eta h$ of every stable gradient step, so on a quadratic the clip never binds while the probes are stable (Proposition 2). When it does bind, the prediction becomes conservative: growing motion is extrapolated at constant speed.

A.4 AdamW form

Assumption 2 (Decoupled AdamW transition)**.** *Every modeled parameter in group p is updated at every reference step with fixed learning rate $\eta_p > 0$ and weight decay $\lambda_p \ge 0$, through its adaptive direction d_n:*

$$\theta_{n+1} = c_p\theta_n - \eta_p d_n, \qquad c_p = 1 - \eta_p\lambda_p. \tag{12}$$

This is decoupled AdamW (Loshchilov & Hutter, 2019). The direction is read back from the parameters,

$$d_n = \big[(\theta_n - \theta_{n+1}) - \eta_p\lambda_p\theta_n\big]/\eta_p, \tag{13}$$

so it already contains the moments, bias correction, and epsilon. The ratio $d_{1,i}/d_{0,i}$ describes adaptive motion, not a Hessian entry.

Theorem 2 (AdamW endpoint decomposition)**.** *Under Assumption 2, let* $R = \mathrm{diag}(r_i)$, $e_n = d_{n+1} - Rd_n$, *and* $T_m(c, R) = \sum_{q=0}^{m-1} c^{m-1-q} R^q$. *Within group* p,

$$\widehat{\theta}_K^{\mathrm{AdamW}} = c_p^{K-1}\theta_1 - \eta_p T_{K-1}(c_p, R)d_1,$$
$$\theta_K - \widehat{\theta}_K^{\mathrm{AdamW}} = -\eta_p \sum_{\ell=1}^{K-2} T_{K-1-\ell}(c_p, R)e_\ell, \tag{14}$$

with norm at most $\eta_p \sum_{\ell=1}^{K-2} T_{K-1-\ell}(|c_p|, \|R\|)\|e_\ell\|$.

Proof. Put $C = c_p I$, $A = -\eta_p I$, and $z_n = d_n$ in Eq. (8); then $M_m = -\eta_p T_m(c_p, R)$. □

Several groups give block-diagonal C and A, which a diagonal R preserves. The quadratic-in-K bound needs $|c_p| \le 1$ and $\max_i |r_i| \le 1$.

A.5 Muon form

For a matrix Θ_n, take the total motion $U_n = \Theta_{n+1} - \Theta_n$, so $C = A = I$. With a scalar trend ρ, $E_n = U_{n+1} - \rho U_n$, and $\Delta_K = \Theta_K - \Theta_0$, Eq. (8) becomes

$$\widehat{\Delta}_K = U_0 + S_{K-1}(\rho)U_1, \qquad \Delta_K - \widehat{\Delta}_K = \sum_{\ell=1}^{K-2} S_{K-1-\ell}(\rho)E_\ell, \tag{15}$$

and the error is at most $\sum_\ell S_{K-1-\ell}(|\rho|)\|E_\ell\|_F$. Muon's orthogonalized motions have nearly equal norm, and then the fitted trend is the cosine between U_0 and U_1. The fitted value near 0.96 in Section 7.3 therefore means consecutive Muon motions are nearly aligned.

B Stochastic, On-Policy, and Clipped Training

RLVR training breaks the clean setting above in five ways: every outer step draws a fresh batch from the current or a recent policy, the clipped surrogate is not smooth, gradients are norm-clipped, the preconditioner is carried across batches, and the gate can switch prediction off. We only assume the training target f is smooth. Lemma 2 bounds the residuals that Appendix A needed, and Theorems 3 and 4 prove convergence.

Protocol. Outer step t starts at (x_t, s_t); $\mathcal{F}_t$ holds everything seen before batch B_t is drawn. On B_t, ℓ_t is the objective of Section 3 with rollouts and advantages fixed, and the supplied gradient is the clipped $\bar{g}_t(\theta) = \Pi_c(\nabla \ell_t(\theta))$, $\Pi_c(v) = \min\{1, c/\|v\|\}v$. The optimizer moves by $\theta \leftarrow \theta - \eta P_t m$, where P_t is a positive diagonal preconditioner read from s_t and held fixed during the step, and $m \leftarrow \beta m + (1-\beta)\bar{g}_t(\theta)$ starts from the stored m_t. The probes use this rule; then m_t is restored and the correction is $m_{t+1} = \beta m_t + (1-\beta)\bar{g}_t(\widetilde{x}_t)$, $x_{t+1} = \widetilde{x}_t - \eta_c P_t m_{t+1}$. For AdamW, $P_t = \mathrm{diag}(1/(\sqrt{\widehat{v}_t} + \epsilon_{\mathrm{opt}}))$. We ignore AdamW's within-step update of $\widehat{v}$, which changes each denominator by less than 0.1% for $\beta_2 = 0.999$ when $\bar{g}_i^2 \le 3v_i$. Clipping gives $1/(c + \epsilon_{\mathrm{opt}}) \le p_{\mathrm{lo}}$ and $p_{\mathrm{hi}} \le 1/\epsilon_{\mathrm{opt}}$. Weight decay is added after Corollary 1.

Assumption 3 (Stochastic protocol)**.** ***(a)*** *$f : \mathbb{R}^d \to \mathbb{R}$ has L-Lipschitz gradient and $f_{\mathrm{inf}} = \inf f > -\infty$.*

(b) *$G_t := \bar{g}_t(x_t)$ satisfies $\mathbb{E}[G_t \mid \mathcal{F}_t] = \nabla f(x_t) + b_t$ and $\mathbb{E}[\|G_t - \mathbb{E}[G_t \mid \mathcal{F}_t]\|^2 \mid \mathcal{F}_t] \le \sigma^2$, with b_t $\mathcal{F}_t$-measurable and $\|b_t\| \le \bar{b}$.*

(c) *For a constant L_B and a random variable $0 \le J_t \le \bar{J}$, $\|\bar{g}_t(\theta) - \bar{g}_t(\theta')\| \le L_B\|\theta - \theta'\| + J_t$ for every pair of points θ, θ' used in step t.*

(d) *$p_{\mathrm{lo}} I \preceq P_t \preceq p_{\mathrm{hi}} I$ with $p_{\mathrm{lo}} > 0$ and $\kappa_P = p_{\mathrm{hi}}/p_{\mathrm{lo}}$; the retention $R_t = \mathrm{diag}(r_{t,i})$, which may be fitted on B_t by Eq. (11), has $r_{t,i} \in [r_{\mathrm{lo}}, 1]$ with $-1 \le r_{\mathrm{lo}} \le 1$ (we use $r_{\mathrm{lo}} = -1$).*

(e) *The gate is constant, $\alpha_t \equiv \alpha \in [0,1]$; one switch to $\alpha = 0$ is treated after Theorem 3.*

(f) *Gradients are clipped at a norm $c < \infty$, the first moment starts at $m_0 = 0$, and $\|\nabla f\| \le G_f$.*

In RLVR, f is the negative expected verifiable reward, which is L-smooth under bounded rewards and bounded derivatives of $\log \pi_\theta$ (Zhang et al., 2020; Agarwal et al., 2021). The bias b_t absorbs group normalization, token averaging, clipping, and staleness, and the one-update baseline carries the same bias. The per-batch loss need not be smooth; the next lemma gives (c) for the clipped surrogate.

Lemma 1 (Clipped-surrogate drift)**.** *Let $\ell_t(\theta) = -\sum_i w_i \min\{\rho_i A_i, \mathrm{clip}(\rho_i, 1-\epsilon, 1+\epsilon) A_i\} + \ell_s(\theta)$ with token weights $w_i \ge 0$ summing to one, advantages A_i, ratios $\rho_i(\theta) = \pi_\theta(o_i)/\pi_{\mathrm{old}}(o_i)$, and smooth extra terms ℓ_s. Let $a_i(\theta) = 0$ when token i is clipped ($A_i > 0, \rho_i > 1+\epsilon$ or $A_i < 0, \rho_i < 1-\epsilon$) and $a_i(\theta) = 1$ otherwise. The supplied gradient is*

$$\nabla \ell_t(\theta) = -\sum_i w_i a_i(\theta) A_i \nabla \rho_i(\theta) + \nabla \ell_s(\theta). \tag{16}$$

If, on a convex set containing the points of step t, $\nabla \rho_i$ is L_i-Lipschitz with $\|\nabla \rho_i\| \le \Gamma_i$ and $\nabla \ell_s$ is L_s-Lipschitz, then Assumption 3(c) holds with

$$L_B = \sum_i w_i |A_i| L_i + L_s, \qquad J_t = \sup_{\theta, \theta'} \sum_{i:\, a_i(\theta) \ne a_i(\theta')} w_i |A_i| \Gamma_i \le 2\kappa_t \max_i |A_i| \Gamma_i, \tag{17}$$

where κ_t is the largest clip fraction $\kappa(\theta) = \sum_i w_i (1 - a_i(\theta))$ among those points. If the batch is on-policy and every point stays within $\log(1+\epsilon)/\Gamma_s$ of x_t, with Γ_s bounding $\|\nabla \log \pi_\theta(o_i)\|$, no token is clipped and $J_t = 0$.

Proof. Each token's term is either $A_i \rho_i$ or a constant, which gives Eq. (16). Between two points, tokens unclipped at both contribute at most $w_i |A_i| L_i \|\theta - \theta'\|$, tokens clipped at both contribute nothing, and tokens clipped at exactly one contribute at most $w_i |A_i| \Gamma_i$. Norm clipping is non-expansive, so the bound carries over to $\bar{g}_t$. For the last claim, the mean-value theorem gives $|\log \rho_i| \le \log(1+\epsilon)$, which keeps ρ_i inside the clip range. □

With token-mean aggregation, $\kappa(\theta)$ is exactly the clip fraction every RLVR trainer already logs (`pg_clipfrac` in VERL), at both probes and the repositioned point. So J_t is a measured quantity, and it is zero in any step whose three passes clip nothing.

Lemma 2 (Derived motion residuals)**.** *Run the reference trajectory of step t from $\theta_0 = x_t$ with the protocol above, with $\|m_t\| \le c$. Let $a = \eta p_{\mathrm{hi}} L_B$ and $\Lambda_t = \|m_t - G_t\|$, the gap between the stored first moment and the fresh gradient. For $0 \le \ell \le K-2$ the residuals $e_\ell = z_{\ell+1} - R_t z_\ell$ satisfy*

$$\|e_\ell\| \le p_{\mathrm{hi}} \big[(1-\beta)\beta^{\ell+1}\Lambda_t + (1-\beta) J_t + (1-\beta^{\ell+1}) ac + (1 - r_{\mathrm{lo}}) c\big], \tag{18}$$

and the second-motion predictor $\widehat{\theta}_K = \theta_1 - \eta S_{K-1}(R_t) z_1$ satisfies

$$\|\theta_K - \widehat{\theta}_K\| \le \eta p_{\mathrm{hi}} \Big[(K-2)\beta^2 \Lambda_t + \tfrac{1}{2}(K-1)(K-2)\big((1-\beta) J_t + \eta p_{\mathrm{hi}} L_B c + (1 - r_{\mathrm{lo}}) c\big)\Big]. \tag{19}$$

Proof. Split $e_\ell = (1-\beta) P_t (\bar{g}_{\ell+1} - m^{(\ell)}) + (I - R_t) P_t m^{(\ell)}$. Every first moment is an average of clipped gradients, so the second part is at most $(1 - r_{\mathrm{lo}}) p_{\mathrm{hi}} c$ and each step moves at most $\eta p_{\mathrm{hi}} c$. Writing $m^{(\ell)}$ as a weighted average of m_t and $\bar{g}_0, \ldots, \bar{g}_\ell$ and applying (c) to each gap gives $\|\bar{g}_{\ell+1} - m^{(\ell)}\| \le \beta^{\ell+1}\Lambda_t + J_t + ac(1 - \beta^{\ell+1})/(1-\beta)$, which is Eq. (18). Summing with the weights $K - 1 - \ell$ of Eq. (9) gives the endpoint bound. □

Every term is an optimizer constant, the retention interval, or something logged during training: the momentum lag Λ_t, the clip fraction inside J_t, and L_B. The probes alone cannot bound the error (Proposition 1); the probes plus smoothness and clipping information can. For SGD on a quadratic, the ratio fit recovers $R = I - \eta H$ and every residual vanishes. Muon's orthogonalization has no uniform Lipschitz constant, so for Muon we get only $\|E_\ell\|_F \le (1 + |\rho|) \max_n \|U_n\|_F$ and report the fitted retention instead (Section 7.3).

Lemma 3 (A PrePO step is a momentum step with a per-coordinate step size)**.** *Under Assumption 3(c)–(f), let $S_t = S_{K-1}(R_t) = \sum_{q=0}^{K-2} R_t^q$, $\gamma_{\mathrm{lo}} = \alpha\eta + \eta_c$, $\gamma_{\mathrm{hi}} = \alpha K\eta + \eta_c$, $a = \eta p_{\mathrm{hi}} L_B$, and $\varrho = \alpha(K-1)/(\alpha+\tau)$ with $\tau = \eta_c/\eta$. Then S_t is diagonal with entries in $[0, K-1]$, and at most 1 wherever $r_{t,i} \le 0$. Moreover,*

$$\begin{aligned} x_{t+1} &= x_t - P_t\big[(\beta\gamma_{\mathrm{lo}} I + E_t)m_t + ((1-\beta)\gamma_{\mathrm{lo}} I + F_t)G_t + W_t\big], \\ m_{t+1} &= \beta m_t + (1-\beta)(G_t + w_t), \\ E_t &= \alpha\eta\beta^2 S_t, \quad F_t = \alpha\eta(1-\beta^2)S_t, \quad W_t = \alpha\eta(1-\beta)S_t\delta_1 + \eta_c(1-\beta)\delta_c, \quad w_t = \delta_c, \end{aligned} \tag{20}$$

where $\delta_1 = \bar{g}_t(\theta_1) - G_t$ and $\delta_c = \bar{g}_t(\widetilde{x}_t) - G_t$ satisfy $\|\delta_1\| \le ac + J_t$ and $\|\delta_c\| \le \alpha K ac + J_t$. Also $\|x_{t+1} - x_t\| \le \gamma_{\mathrm{hi}} p_{\mathrm{hi}} c$, $\|m_{t+1}\| \le c$, and

$$\|W_t + \beta\gamma_{\mathrm{lo}} w_t\| \le \gamma_{\mathrm{lo}} U_t, \qquad U_t := \varrho(1-\beta)(ac + J_t) + \alpha K ac + J_t. \tag{21}$$

Coordinate i moves with total step $\Gamma_{t,i} = \alpha\eta(1 + S_{t,ii}) + \eta_c \in [\gamma_{\mathrm{lo}}, \gamma_{\mathrm{hi}}]$, and its momentum weight lies in $[\beta^2, \beta]$.

Proof. The range of S_t follows from $S_n(r)(1-r) = 1 - r^n \in [0, 2]$ for $r \in [-1, 1)$. The two probes give $z_0 = P_t\bar{m}$ and $z_1 = P_t m^{(1)}$ with $\bar{m} = \beta m_t + (1-\beta)G_t$ and $m^{(1)} = \beta\bar{m} + (1-\beta)(G_t + \delta_1)$, so the predicted move is $-\eta P_t(\bar{m} + S_t m^{(1)})$. Scale it by α, add the correction, and collect the coefficients of m_t, G_t, δ_1, and δ_c; this is Eq. (20). The norm bounds follow because every first moment has norm at most c, and (c) then bounds δ_1 and δ_c. □

So the trend never enters as an error. A coordinate whose motion persists ($r = 1$) takes the full step γ_{hi}, and one whose motion reverses takes at most $2\alpha\eta + \eta_c$. With $R = I$, Eq. (20) is a quasi-hyperbolic momentum step (Ma & Yarats, 2019). This is what separates PrePO from a larger learning rate:

Proposition 2 (Prediction is not a larger step)**.** ***(i)** A learning rate scaled so that one update has step γ_{hi}, or prediction at constant speed ($R = I$, weight prediction), gives every coordinate the step γ_{hi}. PrePO gives coordinate i the step $\alpha\eta(1 + S_{K-1}(r_{t,i})) + \eta_c$, which is γ_{hi} only where $r_{t,i} = 1$ and at most $2\alpha\eta + \eta_c$, for every K, where $r_{t,i} \le 0$. **(ii)** On one coordinate of a quadratic with curvature $h \ge 0$ and plain gradient motion, the fitted trend is $r = 1 - \eta h$ and the predictor returns the true K-step endpoint $r^K\theta_0$. One PrePO step multiplies the coordinate by*

$$\varphi(h) = (1 - \eta_c h)\big(1 - \alpha + \alpha(1-\eta h)^K\big), \qquad \varphi'(0) = -\gamma_{\mathrm{hi}},$$

and $|\varphi(h)| \le 1$ for $0 \le h \le 2/\max(\eta, \eta_c)$. A single step of size γ_{hi} diverges once $h > 2/\gamma_{\mathrm{hi}}$. Weight prediction repositions by $1 - \alpha + \alpha(1-\eta h)(1 - (K-1)\eta h)$, which exceeds 1 once $\eta h > K/(K-1)$, and at $\eta h = \eta_c h = 2$ its full step has magnitude $1 + 2\alpha(K-2)$.

Proof. (i) is Lemma 3 with $S_{K-1}(1) = K - 1$. For (ii), $\eta h S_{K-1}(r) = 1 - r^{K-1}$ makes the prediction exact, and both factors of φ lie in $[-1, 1]$ when $|r| \le 1$. The weight-prediction factor minus one is $\alpha u((K-1)u - K)$ with $u = \eta h$. □

All three updates move flat coordinates at the same speed; they differ on sharp ones. In our RL runs ($\tau = 1$, $\alpha = 0.5$, $K = 5$, so $\gamma_{\mathrm{hi}} = 3.5\eta$), PrePO is stable up to curvature $2/\eta$, one step of size γ_{hi}

only up to $2/(3.5\eta)$, and weight prediction's step reaches magnitude 4 at $2/\eta$. The two probes show which coordinates overshoot, and PrePO shortens exactly those.

The next result turns this per-coordinate step into a rate. It isolates one outer step on a fixed batch, where the batch loss is deterministic, and compares PrePO with one plain gradient step under the Polyak–Łojasiewicz condition (Karimi et al., 2016).

Proposition 3 (Per-batch progress). *Let f be L-smooth with minimum $f^\star$ and satisfy $\frac{1}{2}\|\nabla f(x)\|^2 \geq \mu\,(f(x) - f^\star)$. Take one PrePO step from x with gradient motions ($C = I$, $A = -\eta I$), a diagonal trend clipped to $[-1, 1]$, and $\eta_c = \tau\eta$ with $(1 + 2\tau)L\eta \leq 1$. Let $g = \nabla f(x) \neq 0$, let $\Gamma = \alpha\eta(I + S) + \eta_c I$ with $S = S_{K-1}(R)$ be the per-coordinate step of Lemma 3, and let $\gamma_{\text{eff}} = \langle g, \Gamma g\rangle/\|g\|^2 \in [\gamma_{\text{lo}}, \gamma_{\text{hi}}]$. Then the next point x^+ satisfies*

$$f(x^+) - f^\star \leq \Big(1 - 2\mu\big(\gamma_{\text{eff}} - c_K L\eta^2\big)\Big)\big(f(x) - f^\star\big), \tag{22}$$

with $c_K = (\alpha K + \tau)(1 + 2\tau + 2(\alpha K + \tau))$, while one gradient step of size η gives the factor $1 - 2\mu(\eta - L\eta^2/2)$. Up to terms of order $L\eta^2$, one PrePO step therefore makes the progress of γ_{eff}/η plain steps, between $\alpha + \tau$ and $\alpha K + \tau$.

Proof. The probes give $\theta_1 = x - \eta g$ and $g_1 = \nabla f(\theta_1)$, and Eq. (8) gives $\widehat{\theta} = \theta_1 - \eta S g_1$, so $\widetilde{x} = x - \alpha\eta(g + S g_1)$ and $x^+ = \widetilde{x} - \eta_c \nabla f(\widetilde{x}) = x - \Gamma g - e$ with $e = \alpha\eta S\delta_1 + \tau\eta\delta_c$, $\delta_1 = g_1 - g$, and $\delta_c = \nabla f(\widetilde{x}) - g$. Smoothness gives $\|\delta_1\| \leq L\eta\|g\|$, hence $\|g_1\| \leq 2\|g\|$, $\|\widetilde{x} - x\| \leq \alpha\eta(2K - 1)\|g\|$, and $\|e\| \leq \alpha(1 + 2\tau)KL\eta^2\|g\| \leq (1 + 2\tau)(\alpha K + \tau)L\eta^2\|g\|$, using $0 \leq S \leq K - 1$. The step condition then gives $\|x^+ - x\| \leq 2\gamma_{\text{hi}}\|g\|$. The descent lemma yields $f(x^+) \leq f(x) - \gamma_{\text{eff}}\|g\|^2 + \|g\|\|e\| + 2L\gamma_{\text{hi}}^2\|g\|^2 \leq f(x) - (\gamma_{\text{eff}} - c_K L\eta^2)\|g\|^2$, and the Polyak–Łojasiewicz condition gives Eq. (22). The gradient-step factor is the same argument with $\Gamma = \eta I$ and $e = 0$. □

Coordinates that keep their pace contribute the full step γ_{hi} to γ_{eff}, and coordinates that slow down contribute less, so the rate follows the measured trends. In our RL runs ($\alpha = 0.5$, $\tau = 1$, $K = 5$) one PrePO step does the work of 1.5 to 3.5 plain steps on the batch loss, and Proposition 2 keeps that step stable where a single step of size γ_{hi} is not.

Theorem 3 (Stochastic PrePO with AdamW-type state). *Under Assumption 3, let $\kappa = \beta\gamma_{\text{lo}}/(1 - \beta)$ and $\varepsilon_z = L\kappa p_{\text{hi}} c$, and suppose $64L\gamma_{\text{hi}}^2 p_{\text{hi}} \leq \gamma_{\text{lo}}$. Then*

$$\frac{1}{T}\sum_{t<T}\mathbb{E}\|\nabla f(x_t)\|^2 \leq \frac{4[f(x_0) - f_{\text{inf}}]}{p_{\text{lo}}\gamma_{\text{lo}}T} + A_\sigma\sigma^2 + A_m\overline{\mathcal{M}}_T + A_b\big(\varepsilon_z^2 + \overline{\mathcal{B}}_T\big) + A_c c^2 + C_P\overline{\mathcal{D}}_T, \tag{23}$$

where the bars average over $t < T$ the momentum error $\mathcal{M}_t = \mathbb{E}\|m_t - \nabla f(x_t)\|^2$, the drift $\mathcal{B}_t = \mathbb{E}[\|b_t\|^2 + U_t^2]$, and the preconditioner change $\mathcal{D}_t = \mathbb{E}\|P_{t+1} - P_t\|$ (largest diagonal entry), with

$$A_\sigma = 8\kappa_P\big(L(1 + \varrho)p_{\text{hi}}\gamma_{\text{hi}} + \varrho(1 - \beta^2)\big), \qquad A_m = 8\varrho\beta^2\kappa_P, \qquad A_b = 12(1 + \varrho)\kappa_P,$$
$$A_c = 8L\varrho^2\beta^4\gamma_{\text{lo}}\kappa_P p_{\text{hi}}, \qquad C_P = \frac{4\beta c(G_f + 2L\kappa p_{\text{hi}} c)}{(1 - \beta)p_{\text{lo}}}.$$

The momentum error satisfies

$$\overline{\mathcal{M}}_T \leq \frac{3G_f^2}{(1 - \beta^2)T} + 3\Big(\frac{L\gamma_{\text{hi}}p_{\text{hi}}c}{1 - \beta} + \bar{b} + \alpha K a c + \bar{J}\Big)^2 + \frac{3(1 - \beta)}{1 + \beta}\sigma^2. \tag{24}$$

Proof. Track $z_t = x_t - \kappa P_t m_t$, for which the m_t terms of Lemma 3 cancel:

$$z_{t+1} - z_t = -P_t\big((\gamma_{\text{lo}}I + F_t)G_t + E_t m_t + \gamma_{\text{lo}}u_t\big) - \kappa(P_{t+1} - P_t)m_{t+1}, \qquad \|u_t\| \leq U_t.$$

Apply the descent lemma to $f(z_{t+1})$, telescope over $t < T$, and return to x_t with $\|\nabla f(z_t) - \nabla f(x_t)\| \leq \varepsilon_z$; this gives Eq. (23). The only nonstandard term is the noise times $P_t F_t$, which

has nonzero mean because the trend is fitted on the same batch; it gives the $\varrho(1-\beta^2)$ part of A_σ. Eq. (24) follows by unrolling $m_{t+1}-g_{t+1}=\beta(m_t-g_t)+(g_t-g_{t+1})+(1-\beta)(b_t+\delta_{c,t}+\xi_t)$. □

Gate switches. If the gate switches α to zero once, apply the argument to each phase. The auxiliary point jumps by at most $|\kappa-\kappa'|p_{\mathrm{hi}}c$ at the switch, with $\kappa'=\eta_c\beta/(1-\beta)$, so Eq. (23) holds with $f(x_0)-f_{\inf}$ increased by $G_f|\kappa-\kappa'|p_{\mathrm{hi}}c$ and γ_{lo} replaced by η_c in the denominators.

Corollary 1 (Rate and the price of prediction)**.** *Fix K, α, τ, and β, and set $\eta=\eta_0/\sqrt{T}$. For all sufficiently large T,*

$$\min_{t<T}\mathbb{E}\|\nabla f(x_t)\|^2 \le O\Big(\tfrac{1}{\sqrt{T}}\Big)+28\varrho(1-\beta)\kappa_P\sigma^2+O\big((1+\varrho)^3\kappa_P(\bar b^2+\bar J^2)\big)+C_P\overline{\mathcal{D}}_T.$$

Proof. With $\eta\propto T^{-1/2}$, the step sizes, κ, ε_z, and a are $O(T^{-1/2})$ while ϱ stays fixed. Substituting Eq. (24) into Eq. (23), the σ^2 terms combine to at most $28\varrho(1-\beta)\kappa_P\sigma^2$ and everything else is $O(T^{-1/2})$, a bias or clipping term, or the preconditioner term. □

The baseline ($\alpha=0$) has $\varrho=0$ and the same bias and preconditioner terms, so prediction costs exactly two things. The first is a variance term, because the trend is fitted on the same batch as the gradient. It shrinks with batch size and would vanish for a trend fixed in advance, but such a trend could not see which coordinates overshoot. The second is the clipping term $\bar J$, which is zero whenever no token is clipped. The preconditioner term is zero for SGD with momentum, and at most $C_P d(p_{\mathrm{hi}}-p_{\mathrm{lo}})/T$ for monotone denominators as in AMSGrad.

Weight decay. Our runs use decoupled decay $\lambda=10^{-2}$. The endpoint results include it exactly through $C=(1-\eta\lambda)I$. In the convergence proof, decay adds a shrinkage $-\gamma_{\mathrm{lo}}P_t(\lambda_t P_t^{-1}x_t)$ with $\lambda_t\le(1+\varrho)\lambda$, plus $O(\eta)$ terms, so the theorem holds with U_t increased by $(1+\varrho)\lambda\|(\sqrt{\widehat v_t}+\epsilon_{\mathrm{opt}})\odot x_t\|+O(\eta)$. At $\alpha=0$ this is AdamW's own decay term, which already moves AdamW's stationary points (Zhuang et al., 2022; Xie & Li, 2024); prediction scales it by at most $1+\varrho$.

Reading the constants. Like other bounds for adaptive methods (Chen et al., 2019; Défossez et al., 2022), Eq. (23) is a worst case and does not predict the loss at LLM scale; κ_P alone can reach $(c+\epsilon_{\mathrm{opt}})/\epsilon_{\mathrm{opt}}$. Its content is comparative: L, κ_P, σ, and $\bar b$ enter the $\alpha=0$ bound the same way, and the extra terms are either explicit in (α,K,τ,β) or logged during training. Quantitative claims about the prediction rest on measured endpoint errors instead, which sit a factor of 1.2 to 6 below the bound (Table 6).

Muon. For matrix blocks $\Theta=(\Theta_b)$, let $\|X\|_{\mathrm{op}}=\max_b\|X_b\|_{\mathrm{op}}$, with dual norm $\|X\|_*=\sum_b\|X_b\|_*$, and $\bar r=\sum_b\min(m_b,n_b)$, so $\|X\|_*\le\sqrt{\bar r}\|X\|_F$. Muon keeps $M\leftarrow\mu M+(1-\mu)\bar g$ and moves each block by $-\eta\,\mathrm{polar}(M_b)$, the steepest-descent direction in $\|\cdot\|_{\mathrm{op}}$ (Bernstein & Newhouse, 2024). Since $\|\mathrm{polar}(M)\|_{\mathrm{op}}\le1$ and $\langle M,\mathrm{polar}(M)\rangle=\|M\|_*$, for every Y

$$\langle Y,\mathrm{polar}(M)\rangle\ge\|M\|_*-\|Y-M\|_*\ge\|Y\|_*-2\|Y-M\|_*. \tag{25}$$

PrePO predicts with the motions $U_j=-\eta\,\mathrm{polar}(M^{(j)})$ and a scalar trend $\rho_b\in[-1,1]$ per block, then corrects with $\Theta_{t+1}=\widetilde\Theta_t-\eta_c\,\mathrm{polar}(M_{t+1})$, where $M_{t+1}=\mu M_t+(1-\mu)\bar g_t(\widetilde\Theta_t)$.

Theorem 4 (Stochastic PrePO with Muon)**.** *Assume (i) $\|\nabla f(X)-\nabla f(Y)\|_*\le L_{\mathrm{op}}\|X-Y\|_{\mathrm{op}}$ and $f_{\inf}>-\infty$; (ii) $G_t=\bar g_t(\Theta_t)=\nabla f(\Theta_t)+b_t+\xi_t$ with $\mathbb{E}[\xi_t\mid\mathcal{F}_t]=0$, $\mathbb{E}\|\xi_t\|_F^2\le\sigma^2$, and $\|b_t\|_*\le\bar b$; (iii) $\|\bar g_t(\theta)-\bar g_t(\theta')\|_*\le L_B^{\mathrm{op}}\|\theta-\theta'\|_{\mathrm{op}}+J_t$ for the points used in step t, with $\mathbb{E}J_t\le\bar J$; and (iv) a constant gate. Let $\gamma=\alpha K\eta+\eta_c$, $\gamma_{\mathrm{lo}}=\alpha\eta+\eta_c$, and $\varepsilon_0=(1-\mu)G_0-\nabla f(\Theta_0)$. Then*

$$\begin{aligned}\frac{1}{T}\sum_{t<T}\mathbb{E}\|\nabla f(\Theta_t)\|_* \le{}& \frac{f(\Theta_0)-f_{\inf}}{\gamma_{\mathrm{lo}}T}+\frac{L_{\mathrm{op}}\gamma^2}{2\gamma_{\mathrm{lo}}}+\frac{\gamma}{\gamma_{\mathrm{lo}}}\Big[\frac{4\,\mathbb{E}\|\varepsilon_0\|_*}{(1-\mu)T}+\frac{4\mu L_{\mathrm{op}}\gamma}{1-\mu}\\ &+6\sqrt{\bar r}\sigma\sqrt{1-\mu}+6\bar b+4(1+\alpha K)L_B^{\mathrm{op}}\eta+4\bar J\Big].\end{aligned} \tag{26}$$

With $1-\mu = T^{-1/2}$ and $\eta \propto T^{-3/4}$ the right side is $O(T^{-1/4}) + (\gamma/\gamma_{\mathrm{lo}})(6\bar{b} + 4\bar{J})$, the rate of normalized SGD with momentum (Cutkosky & Mehta, 2020) and of Muon itself ($\alpha = 0$).

Proof. Let $g_t = \nabla f(\Theta_t)$ and $\varepsilon_t = M^{(0)} - g_t$ for the first probe's moment. Every block moves by at most γ with coefficient at least γ_{lo}, so the descent inequality and Eq. (25) give

$$f(\Theta_{t+1}) \le f(\Theta_t) - \gamma_{\mathrm{lo}}\|g_t\|_* + 4\gamma\|\varepsilon_t\|_* + 2\gamma(1-\mu)\big(\|b_t + \xi_t\|_* + \|\delta_1\|_* + \|\delta_c\|_*\big) + \tfrac{L_{\mathrm{op}}}{2}\gamma^2.$$

Bounding $\mathbb{E}\|\varepsilon_t\|_*$ from its momentum recursion and summing over $t < T$ gives Eq. (26). □

For Muon the trend enters only through $\gamma/\gamma_{\mathrm{lo}} = 1 + \varrho$, because normalized motions keep every predicted step bounded, and the same-batch fit adds no separate noise term. Lemma 1 gives (iii) with $L_B^{\mathrm{op}} = \bar{r}L_B$ and $\sqrt{\bar{r}}J_t$ in place of J_t. An approximate orthogonalizer such as Newton–Schulz, returning O with $\|O\|_{\mathrm{op}} \le \omega_{\mathrm{hi}}$ and $\langle M, O\rangle \ge \omega_{\mathrm{lo}}\|M\|_*$, changes only the constants: γ_{lo} becomes $\omega_{\mathrm{lo}}\gamma_{\mathrm{lo}}$, the factor 2 becomes $\omega_{\mathrm{lo}} + \omega_{\mathrm{hi}}$, and γ becomes $\omega_{\mathrm{hi}}\gamma$ in the smoothness and drift terms.

C Additional Full-Parameter and Generality Results

C.1 Primary Configurations

Table 4 summarizes the main training and evaluation settings for RLVR, SFT, and OPD. The PrePO settings K, α, and η_c were fixed before any evaluation, and none was tuned on the reported benchmarks. Every reported score comes from the final checkpoint of its run.

C.2 Optimizer and K–α Ablations

Figure A1 tracks accuracy over 300 steps for AdamW and Muon at $K \in \{3, 5\}$ and the α values shown in the legends. At $K = 3$, $\alpha = 0.5$ finishes highest for both optimizers: AdamW improves steadily at this setting, while Muon's $\alpha = 0.5$ curve finishes above its other plotted settings, including $\alpha = 0.8$. At $K = 5$, the ordering depends on the optimizer. AdamW's $\alpha = 0.5$ curve makes its strongest gains later in training and finishes highest in its panel; Muon's $\alpha = 0.3$ curve rises more steadily and finishes highest, while its $\alpha = 0.5$ and $\alpha = 0.1$ curves remain lower and vary more over the run. Comparing horizons, the $\alpha = 0.3$ curves finish higher at $K = 5$ than at $K = 3$ for both optimizers, whereas $\alpha = 0.5$ finishes a little higher for AdamW and lower for Muon. This pattern is consistent with an interaction between the optimizer, prediction horizon, and the fraction of the predicted displacement that is applied. In PrePO, K determines the virtual horizon and hence the predicted endpoint $\hat{\theta}^{t,K}$, while α sets the fraction of that displacement used to reposition the parameters, $\tilde{\theta}^t - \theta^t = \alpha(\hat{\theta}^{t,K} - \theta^t)$. Thus, increasing α scales the move for a fixed endpoint, but changing K changes the endpoint itself; the curves show why the two choices should be read together. The strongest setting in this sweep varies by optimizer and horizon, and the trajectories also show distinct paths to their final values, with steady gains in some settings and late improvement or fluctuation in others.

C.3 Looking Further Ahead Without Paying for It

Two beliefs make multi-step methods look expensive and fragile. The first is that looking K steps ahead must cost about K steps of compute. Figure A2(a) shows why it does not here. SFPO pays $K+1$ backward passes per update, one for each real step and one for the correction, while PrePO pays three whatever K is, and the logged runs confirm the count at $K = 5$ and at $K = 10$, where a predicting step costs 3.0 times a plain one. At $K = 10$ that is 3.7 times fewer backward passes than SFPO. The second belief is that the path of an LLM optimizer is too irregular to foresee from two steps. Running the real K steps next to the prediction on the same batch shows otherwise. The predicted move points the same way as the real one, with a cosine of 0.993 for AdamW and 0.998

Table 4: Primary training and evaluation configurations.

Item	Setting
	RLVR
Training pool	17,917 DAPO-Math-17K prompts and 7,500 Hendrycks MATH training prompts (25,417 total), serialized in LightEval format (Yu et al., 2025; Hendrycks et al., 2021).
Pool sampling	all_probabilistic; p_{easy}=0.75, p_{hard}=0.5; refill each batch to its target size.
Models	Qwen2.5-Math-1.5B-Instruct, Qwen2.5-Math-7B-Instruct, Qwen2.5-3B-Instruct, and Llama-3.2-3B-Instruct (Yang et al., 2024; Qwen et al., 2024; Meta AI, 2024).
Training stack	VERL with FlashInfer (Sheng et al., 2025; Ye et al., 2025).
Context and generation limits	4096-position context; 1024-token prompt limit; 3,072-token response limit in training and evaluation.
Training sampler	Temperature 1.0; top-p 1.0.
Learning rate	10^{-6}.
Optimizer and loss settings	AdamW; gradient clipping 1.0; policy clipping 0.2; entropy coefficient 0.001; no KL penalty.
Main-run budget	Train/minibatch sizes 256/64; n=8 responses per prompt; 400 steps; PrePO uses K=5, α=0.5 for AdamW and Muon; entropy gate as in SFPO (Wang et al., 2026b) with threshold 1 and patience 1.
Time to target	Target is GRPO's peak greedy Pass@1, averaged over the six benchmarks, over its checkpoints; steps and wall-clock time are recorded at the first checkpoint that reaches it.
Hardware and timing	4× H100 GPUs; wall-clock time is the elapsed time of full training steps, including rollout generation.
Muon sweep	Per arm, $K \in \{3, 5, 10\}$ and $\alpha \in \{0.1, 0.3, 0.5, 0.8\}$.
RL ablation budget	Train/minibatch sizes 64/16; 300 steps; gate with threshold 1, patience 1, and warmup 30.
	SFT and OPD
SFT dataset	Public K-and-K knights-and-knaves set (Xie et al., 2025), converted to prompts, step-by-step reasoning, and canonical JSON answers.
SFT split	3,600 training examples (900 each for 3–6 people); 400 held-out; 400 IID test; 200 OOD test examples for 7–8 people.
SFT models and budget	Qwen2.5-1.5B, Llama-3.2-3B, and Qwen2.5-7B; batch size 256; 400 steps.
SFT learning rate	10^{-5}.
OPD models and budget	Qwen2.5-1.5B, Qwen2.5-3B, and Qwen2.5-7B; batch size 256; n=1; 100 steps.
OPD objective	On-policy distillation (Agarwal et al., 2024): per-token reward $\log \pi_{\text{T}}(y_t) - \log \pi_{\text{old}}(y_t)$ on the student's own samples, centered by a top-128 student baseline and scaled by 0.1; teacher Qwen2.5-Math-7B-Instruct for 1.5B/3B students, and Qwen2.5-32B-Instruct for the 7B student.
OPD learning rate	5×10^{-6}.
SFT and OPD PrePO setting	Batch size 256; K=5, α=0.3.
	Evaluation
Benchmarks	MATH-500 (500), AIME24/25 (30 each), AMC23 (83), MinervaMATH (272), and OlympiadBench (674).
Decoding and metric	Greedy decoding (temperature 0), one response per problem; Pass@1 is computed for each benchmark. Avg. and Overall then average the six test columns.
Benchmark reporting	Scores at the final checkpoint of each run.

Table 5: Rollouts and generated tokens needed to reach the baseline target. Shaded ratios divide the baseline's value by each method's value. †Scaled linearly from the step-100 count.

Method	Steps ↓	Generated tokens ↓	Rollouts ↓	Rollout ratio ↑	Token ratio ↑
Qwen2.5-Math-1.5B, AdamW					
GRPO	392	738,973,428	802,816	1.00×	1.00×
SFPO	204	382,022,780	417,792	1.92×	1.93×
PrePO	**113**	**212,310,988**	**231,424**	**3.47**×	**3.48**×
Llama-3.2-3B, AdamW					
GRPO	285	378,267,892	583,680	1.00×	1.00×
SFPO	135	184,979,928	276,480	2.11×	2.04×
PrePO	**90**	**128,451,916**	**184,320**	**3.17**×	**2.94**×
Qwen2.5-Math-7B, AdamW					
GRPO	387	783,022,193	792,576	1.00×	1.00×
SFPO	107	242,768,854	219,136	3.62×	3.23×
PrePO	**79**	**185,340,191**	**161,792**	**4.90**×	**4.22**×
Qwen2.5-Math-1.5B, Muon					
GRPO	345	648,502,680	706,560	1.00×	1.00×
PrePO	**115**	**224,500,002**	**235,520**	**3.00**×	**2.89**×
Llama-3.2-3B, Muon					
GRPO	255	341,630,696	522,240	1.00×	1.00×
PrePO	**90**	**128,389,664**	**184,320**	**2.83**×	**2.66**×
Qwen2.5-Math-7B, Muon					
GRPO	340	688,272,384	696,320	1.00×	1.00×
PrePO	**104**	**214,842,560**	**212,992**	**3.27**×	**3.20**×
Qwen2.5-1.5B, OPD					
OPD	95	29,312,013	24,320	1.00×	1.00×
PrePO	**20**	**5,607,440**	**5,120**	**4.75**×	**5.23**×
Qwen2.5-3B, OPD					
OPD	65	19,795,125†	16,640	1.00×	1.00×
PrePO	**34**	**10,042,332**	**8,704**	**1.91**×	**1.97**×
Qwen2.5-7B, OPD					
OPD	52	16,240,640	13,312	1.00×	1.00×
PrePO	**26**	**7,808,000**	**6,656**	**2.00**×	**2.08**×

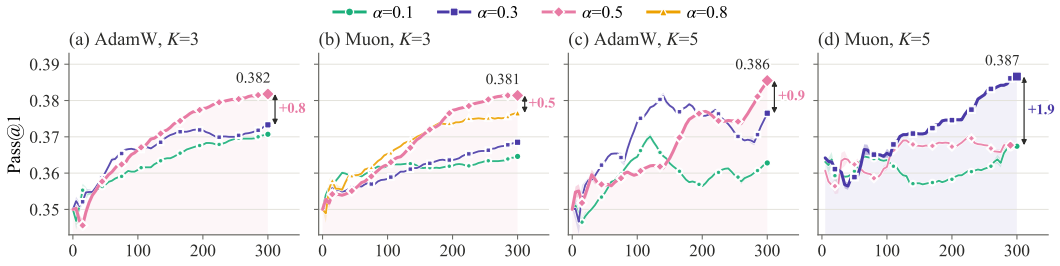

Figure A1: Greedy Pass@1, averaged over the six benchmarks, over training steps for AdamW and Muon at $K = 3$ and $K = 5$. Each colour is one α. The shaded curve finishes highest in its panel; its final score is labelled, and the arrow gives its lead over the next α in points.

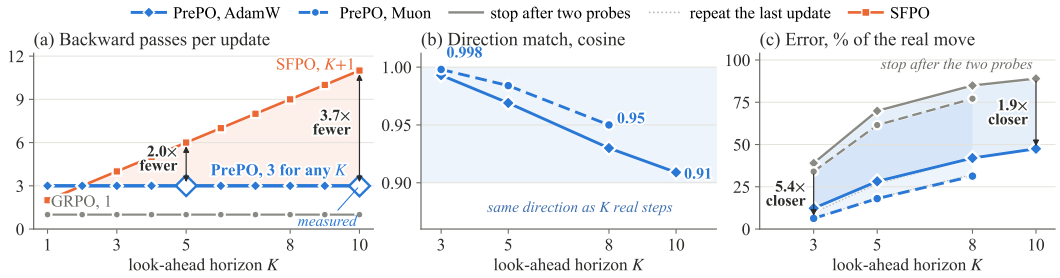

Figure A2: (a) Backward passes per minibatch update against the look-ahead horizon K; open diamonds are measured from the Muon $K = 5$ and AdamW $K = 10$ runs. (b, c) Qwen2.5-Math-1.5B with $\alpha = 0.8$, with the real K steps run beside the prediction on the same batch: medians over training of the cosine between the predicted and real moves, and of the endpoint error as a share of the real move.

for Muon at $K = 3$ that stays above 0.9 even at $K = 10$, and it lands two to five times closer to the real endpoint than stopping after the two probes. The remaining error grows with K, which is why PrePO moves only a fraction α of the way and then takes one real corrective step, and why the runs in Figure A3 favour a smaller α at long horizons. The path is smooth enough that repeating the last update lands in almost the same place, and that smoothness is exactly what lets a two-step prediction see this far. Repetition matches PrePO here because the fitted trends sit near one. The two differ on coordinates whose motion slows or reverses. On a quadratic, repetition keeps the full step there and diverges at high curvature, while PrePO shortens it (Proposition 2).

C.4 What Prediction Costs and What α Changes

A training step splits its batch into minibatches and takes one optimizer update on each, so with 256 prompts per step and minibatches of 64 there are four updates per step. A plain update pays one backward pass, SFPO pays K or $K+1$, and PrePO pays three, two for the probes and one for the correction, so at $K = 5$ a step costs four backward passes for GRPO, twelve for PrePO, and twenty or more for SFPO. Figure A3 checks this count in Muon runs at $K = 5$ and AdamW runs at $K = 10$, which use 64 prompts and minibatches of 16 and therefore also take four updates per step. Each minibatch is further split into micro-batches by a token budget, so the logged count moves with response length, yet while PrePO is predicting the count is always a multiple of three, and it falls from about 17.1 to 5.7 per step, a ratio of 3.0, once the entropy gate hands training back to the plain optimizer. The gate closes between steps 82 and 88 for every α, so prediction is used during the early phase where it saves the most steps, and over 300 steps the runs use about half the backward

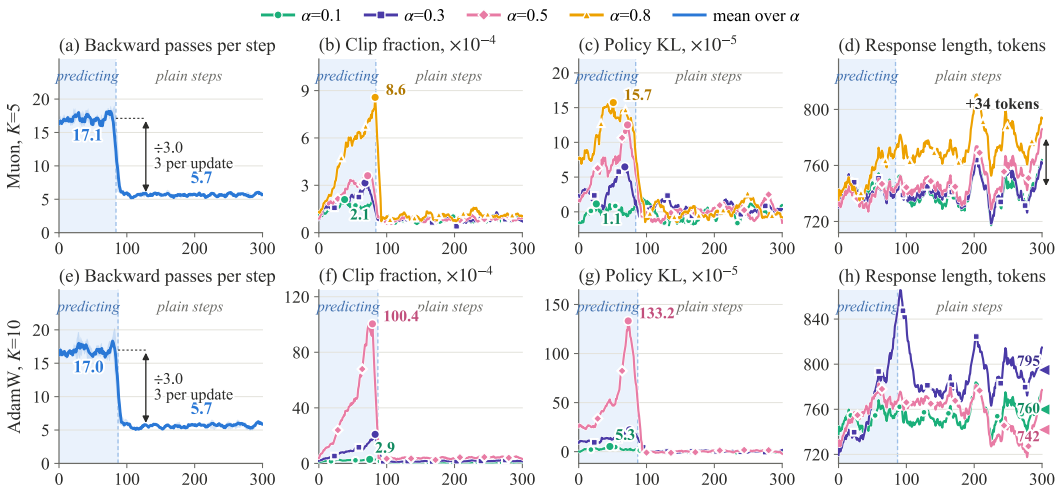

Figure A3: Muon at $K = 5$, top, and AdamW at $K = 10$, bottom, over training steps. Shading marks where PrePO predicts before the entropy gate turns it off; panels (a) and (e) show the mean over α with its range. Moving averages span 9 steps, and 21 for KL and length.

passes they would if prediction stayed on throughout. The value of α leaves this cost unchanged and instead sets how far each corrected update lands from the current policy. A larger α raises the policy KL and the clip fraction while prediction is on, and the clip fraction peaks at 2.1, 3.2, 3.6, and 8.6×10^{-4} for $\alpha = 0.1$, 0.3, 0.5, and 0.8, so even the largest setting clips fewer than one token in a thousand. Both signals return to the same level for every α once the gate closes. Response length follows the same order and keeps it. All four runs start near 740 tokens, and over the last 50 steps they average 745, 746, 761, and 779 tokens, so a larger α moves the policy further along the direction in which RLVR is already lengthening its reasoning, and that difference remains after prediction has stopped. The AdamW runs at $K = 10$ in the lower row repeat the cost story at twice the horizon. Doubling K leaves every update at three backward passes, 17.0 against 5.7 per step, and the gate again closes near step 87. What changes is how strongly α matters. At this horizon the predicted endpoint lies further out, so the same α moves the policy much more, and the clip fraction peaks at 2.9, 20.9, and 100.4×10^{-4} for $\alpha = 0.1$, 0.3, and 0.5, about one token in a hundred at the largest setting, before it falls back once prediction stops. Response length does not follow α here and ends at 760, 795, and 742 tokens, so the lengthening seen with Muon depends on the optimizer and the horizon rather than on α alone. Read together, the two rows favour a smaller α when the horizon is long.

C.5 Implementation Fidelity and Bound Validation

To check the implemented predictor against the finite-horizon analysis, Table 6 compares the realized endpoint error, $\|\theta^{t,K} - \widehat{\theta}^{t,K}\|$, with the corresponding theoretical bound across the listed models, optimizers, K, and α settings. The measured error is below the reported bound in every row. Both generally increase with K: for Qwen2.5-Math-1.5B with AdamW and $\alpha = 0.8$, the error rises from 0.0029 at $K = 3$ to 0.0127, 0.0363, and 0.0555 at $K = 5, 8, 10$, while the corresponding bounds rise from 0.0037 to 0.0288, 0.101, and 0.174. Across the matched model, K, and α settings shown, Muon has lower measured endpoint error than AdamW; at $K = 5$ on the 1.5B model, reducing α from 0.8 to 0.3 also lowers the measured error for both optimizers. Bounds marked * in the table are estimated, and the rest are logged. Together, these measurements provide an empirical check that the implemented predictions remain within the reported bounds across the tested settings, consistent with the finite-horizon analysis in Section 6.

The settings in Table 6 are deliberately harder than the ones PrePO trains with. The main runs use $K = 5$ and $\alpha = 0.5$, while the table reaches $K = 10$ and $\alpha = 0.8$, and even there the bound stays above the measured error by a margin of 1.2 to 6.0, so the prediction never leaves the region the analysis guarantees. The check also carries across model size. From the 1.5B to the 3B model at $K = 3$ and $\alpha = 0.8$ the error grows, but the bound grows with it, and the error stays near four fifths of the bound for both optimizers. The bound has no term in the number of parameters; size enters only through the leftovers measured on the same model, which is why it keeps pace as the model grows.

Table 6: Endpoint error against the bound of Eq. 4. Error is $\|\theta^{t,K} - \widehat{\theta}^{t,K}\|$, the distance between the predicted endpoint and the endpoint of K real updates on the same batch, in the Euclidean norm (Frobenius for Muon matrices). Bound is the right side of Eq. 4 with ε_e the largest measured leftover $\|e_j\|$. The margin is the bound over the error. *Estimated rather than logged.

		AdamW			**Muon**		
K	α	Bound	Error	Margin	Bound	Error	Margin
Qwen2.5-Math-1.5B							
3	0.8	0.0037	**0.0029**	1.3×	0.0012	**0.0010**	1.2×
5	0.8	0.0288	**0.0127**	2.3×	0.0066	**0.0047**	1.4×
5	0.3	0.0286	**0.0048**	6.0×	0.0066	**0.0018**	3.7×
8	0.8	0.101*	**0.0363**	2.8×	0.023*	**0.0128**	1.8×
10	0.8	0.174*	**0.0555**	3.1×		–	
Qwen2.5-3B							
3	0.8	0.0058	**0.0046**	1.3×	0.0022	**0.0017**	1.3×
5	0.5	0.042*	**0.0115**	3.7×	0.0105*	**0.0046**	2.3×

C.6 Fallback Mechanism and Stability Ablation

PrePO employs an entropy gate that triggers a fallback when output distribution entropy exhibits a sharp contraction. In our default configuration, this fallback is *permanent*: once triggered, the repositioning coefficient α is locked to 0 for the remainder of training, and the model updates solely via standard base optimizer steps. To evaluate this design choice, we compare three distinct fallback regimes on Qwen2.5-Math-1.5B with Muon over a 300-step training budget: (1) **No fallback**, where predictive repositioning remains active across all 300 steps regardless of entropy dynamics; (2) **Temporary fallback**, where predictive extrapolation is suspended for a single update ($\alpha = 0$) whenever the entropy gate fires, but re-engages ($\alpha = 0.3$) on subsequent steps if entropy stabilizes; and (3) **Permanent fallback (PrePO)**, where triggering the entropy gate permanently transitions training to standard optimizer updates ($\alpha = 0$) for all remaining steps.

Table 7 reports evaluation results across six mathematical reasoning benchmarks. Permanent fallback has the highest mean accuracy, 38.88, ahead of temporary fallback at 37.70 by 1.18 points and no fallback at 31.07 by 7.81 points.

Table 7: Ablation of the fallback mechanism on Qwen2.5-Math-1.5B with Muon over 300 steps. No fallback continuously extrapolates across all updates; Temporary fallback suppresses extrapolation for single steps upon entropy gate triggers; Permanent fallback (PrePO) permanently reverts to standard optimizer updates once the entropy gate fires.

Benchmark	No fallback	Temporary fallback	Permanent fallback (PrePO)
Mean across benchmarks	31.07	37.70	**38.88**
MATH-500	70.60	**77.20**	75.40
AIME 2024	3.33	10.00	**13.33**
AIME 2025	10.00	**16.67**	**16.67**
AMC 2023	42.17	**56.63**	55.42
Minerva Math	22.79	26.10	**29.41**
OlympiadBench	37.54	39.61	**43.03**

Why the gate helps. Early in training, up to about step 85, entropy is high and consecutive motions point the same way; the fitted Muon trend stays near 0.96 (Figure 4a). Here the two-probe prediction is accurate and PrePO saves the most steps. Later, entropy falls sharply (Figure 4b) and the model settles into sharper regions, where the leftover term ε_e of Eq. 4 is no longer small compared with the region it must stay in.

No fallback. Without the gate, PrePO keeps predicting through this late phase. Small prediction errors now push the repositioned point past the region the policy should stay in, the clipped surrogate discards more tokens, and accuracy declines by step 300. The mean falls to 31.07, and AIME 2024 drops to 3.33.

Temporary fallback. Pausing prediction for one step avoids this collapse but does not settle training. The pause steadies the policy, prediction restarts, the next predicted move overshoots, and the gate fires again, over 130 times in 300 steps. This on-off pattern keeps disturbing the optimizer's moment estimates. The mean reaches 37.70 but trails permanent fallback on the hardest benchmarks, for example 39.61 against 43.03 on OlympiadBench.

Permanent fallback. Treating the first gate event as the end of the prediction phase keeps the benefit of both regimes. PrePO predicts while the path is regular, then the plain optimizer finishes without jumps in its state. It also costs less, because backward passes fall from 17.1 to 5.7 per step once prediction stops (Appendix C.4). It has the best mean, 38.88, and the best or tied-best score on four of the six benchmarks.

C.7 Code Generation

This run is a compact check that the gain holds outside math, alongside the logic puzzles of SFT and the distillation runs, rather than a full code benchmark. We train Qwen2.5-Coder-1.5B (Hui et al., 2024) with full-parameter RLVR on KodCode (Xu et al., 2025) and evaluate on HumanEval+ (Chen et al., 2021; Liu et al., 2023). GRPO and PrePO share a constant learning rate of 10^{-6} and no KL loss. PrePO uses $K = 10$ and $\alpha = 0.5$, with the remaining settings as in the main runs. We report greedy evaluation with Pass@1. The two runs end within one point of each other, so we compare the time to reach 90% of GRPO's final pass@1.

Table 8 shows that PrePO leads at every checkpoint. The lead is largest early in training, 62.09 against 54.47 at step 10 and 66.82 against 55.44 at step 20. PrePO reaches the target with $3.4\times$ less wall-clock time and ends slightly higher, 71.06 against 70.15 at step 190. The gain therefore carries from math reasoning to code.

Table 8: Code generation with Qwen2.5-Coder-1.5B trained on KodCode and evaluated on HumanEval+. Scores use greedy evaluation (Pass@1). The last column is the wall-clock speedup in reaching 90% of GRPO's final pass@1.

Method	Step 10	Step 20	Step 80	Step 100	Step 190	Wall-clock speedup ↑
GRPO	54.47	55.44	63.62	65.07	70.15	1.0×
PrePO	**62.09**	**66.82**	**68.98**	**70.91**	**71.06**	**3.4**×