EXACT: OPTIMIZING ADAPTIVE INFERENCE UTILITY VIA EXECUTION COMPILATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Training for adaptive inference requires assigning credit to responses for their effects on terminal answer quality and acquisition cost. We introduce EXACT, which computes training signals for the expected utility of a specified adaptive procedure. Given IID response banks and controllers with a finite sufficient-state representation, it averages over subset selection, reveal order, and stopping decisions by reusing shared states. Continuation values and forward reach probabilities yield causal credits for the original responses. These credits weight generator sequence scores with the controller fixed (EXACT-G); differentiating the same compiled value trains the controller with the generator fixed (EXACT-C). The generator estimator exactly averages its causal trajectory counterpart conditional on the sampled bank, retaining response-sampling variation. Across three paired training seeds for Qwen3-4B under a 200-update budget, development-selected EXACT-G checkpoints improve mean deployment utility across six benchmarks by 0.0140 and adaptive accuracy by 1.19 percentage points over correctness GRPO, with 1.54% fewer expected rollouts. A separate Qwen3-1.7B rank-eight LoRA diagnostic attributes 51% of sampled-gradient second-moment energy to execution variation under aggressive stopping. On frozen Qwen3-4B banks, EXACT-C computes exact conditional controller gradients and achieves mean held-out utility comparable to Order-MC-64 with 19.5% less controller-training CPU time.

1 INTRODUCTION

When inference can stop early, a response matters through the computation that uses it. It may establish agreement, break a tie, or affect whether another candidate is acquired. Self-consistency aggregates sampled reasoning paths (Wang et al., 2023); adaptive variants decide when enough evidence has arrived (Aggarwal et al., 2023; Li et al., 2024). Training for this procedure must account for both terminal answer quality and acquisition cost.

Individual-response correctness alone does not capture this deployment utility. Set-level objectives account for interactions among responses (Orney et al., 2026); an adaptive procedure additionally decides which responses to reveal and when to stop. Trajectory policy gradient can optimize its utility by sampling executions and assigning revealed responses their remaining return. The computational question is how to average those credits over the supported executions, conditional on sampled responses.

We introduce EXACT (EXecution-Averaged Compilation for Training), which computes these training signals by compiling adaptive executions (Figure 1). Given an IID response bank, a controller, and a utility, the execution distribution is known. For a count-based controller, histories with the same revealed answer counts share the same continuation problem. A dynamic program reuses these states to average over subset selection, reveal order, and stopping decisions without enumerating paths. Model responses remain sampled; compilation averages over the controller's supported executions.

The compiler preserves response-level credit despite sharing state-level computation. Backward continuation values and forward reach probabilities assign an execution-averaged causal credit to each original response. With the controller fixed, EXACT-G uses these credits to weight language-model sequence scores. With the generator fixed, EXACT-C differentiates the same compiled value to train the stopping policy. These are two optimization interfaces to one deployment objective, with additional controller memory retained whenever counts alone are insufficient.

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.

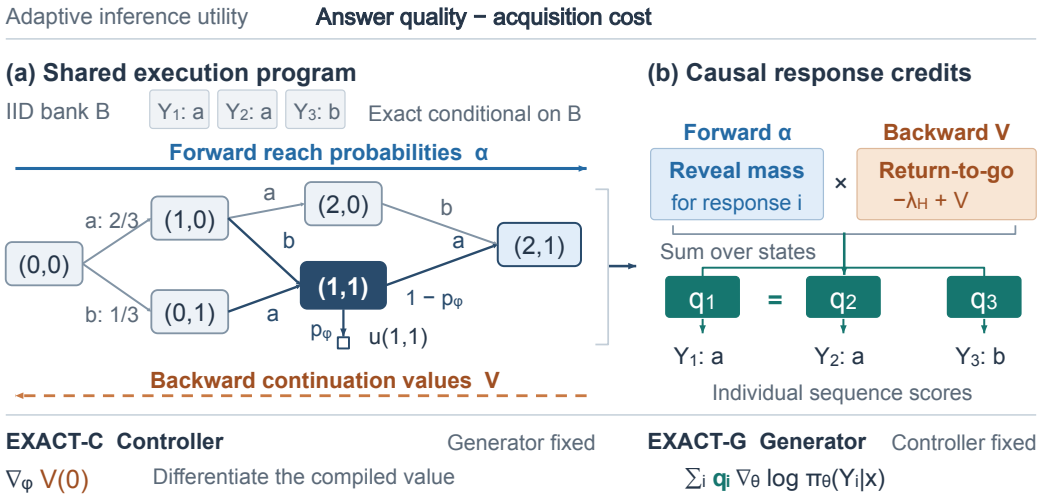

Figure 1: **Shared execution computation preserves response-level credit.** (a) For the illustrated IID bank with answers a, a, b and $N = H = 3$, different reveal orders merge at the same count state. Forward reach probabilities and backward continuation values reuse this shared program. Edges label acquired answers; stopping is illustrated only at $(1, 1)$. (b) Each response's causal credit sums its reveal mass times the remaining return, including acquisition cost. Responses in the same answer class share coefficients ($q_1 = q_2$), while individual sequence scores remain distinct. EXACT-G weights these scores with the controller fixed; EXACT-C differentiates $V(\mathbf{0})$ with the generator fixed. Exactness is conditional on the sampled bank; extra controller memory is retained when needed.

Across three paired training seeds for Qwen3-4B, EXACT-G improves mean deployment utility and adaptive accuracy with fewer expected rollouts than GRPO with per-response correctness rewards on six mathematical benchmarks. This evaluates the complete deployment-aware training method. A separate Qwen3-1.7B diagnostic measures execution error in real generator-gradient geometry. On frozen Qwen3-4B banks, EXACT-C provides exact conditional gradients with lower measured controller-training CPU time than Order-MC-64 and comparable mean held-out utility.

2 OPTIMIZING ADAPTIVE INFERENCE UTILITY

For a prompt x, let $\pi_\theta(\cdot \mid x)$ denote the complete response-sampling policy, including its fixed decoding temperature. A deployment controller acquires at most $H \geq 1$ responses. Each acquisition draws $Y_t \sim \pi_\theta(\cdot \mid x)$ independently, and an extractor maps the response to an answer class $z_t = g(Y_t)$. The controller updates a state $s_t = T(s_{t-1}, z_t)$ and either stops or acquires another response. Its stop probability is $p_\phi(s_t)$; deterministic rules are the special case $p_\phi \in \{0, 1\}$. The first acquisition is forced, $p_\phi(s_0) = 0$, and stopping is forced at H.

We assume a finite sufficient state for future decisions and terminal utility. In the count-state instance, the revealed answer histogram is sufficient; rules that distinguish histories with the same histogram require additional controller memory. The controller only uses revealed information and does not alter the sampling law of future responses. Correctness labels evaluate the terminal answer but are never controller observations.

If $\tau \leq H$ is the stopping time, $u(s_\tau)$ the terminal quality, and $\lambda_H = \beta/H$ the cost per acquired response, the deployment objective is

$$J_H(\theta, \phi) = \mathbb{E}[u(s_\tau) - \lambda_H \tau]\,. \tag{1}$$

The expectation includes prompts, generated responses, and controller actions. Our experiments use correctness of the terminal plurality answer for u and $\beta = 0.2$. A deployment profile combines several fixed controllers and horizons with nonnegative weights w_m summing to one:

$$J_{\mathcal{P}}(\theta) = \sum_{m \in \mathcal{P}} w_m J_{H_m}(\theta, \phi_m). \tag{2}$$

Compilation and generator credits apply termwise to any such profile built from supported operators. Our generator experiment averages six terms—aggressive, medium, and conservative controllers at $H \in \{8, 16\}$—as one instantiation. Controller training instead optimizes one learned stopping policy at a specified (H, β).

The two training settings optimize these deployment objectives with the other parameter block fixed. Generator training uses compiled causal credits under the fixed profile to weight response sequence scores. Controller training differentiates conditional deployment values over frozen response banks. Figure 1 connects the deployment procedure to both signals; the compiler below computes the shared values and response credits.

Pool augmentation. For training or evaluation, draw an IID bank (the *superpool*) $B = (Y_1, \ldots, Y_N)$ with $N \geq H$, then reveal a uniform random permutation of its indices until stopping or reaching H. Every prefix has the same unconditional law as online IID generation. This separates response randomness, contained in B, from execution randomness ξ: reveal order, the induced size-H subset when $H < N$, and stochastic controller actions. We compile the expectation over ξ conditional on B. Pool augmentation is an analysis and training device; online deployment need not generate candidates that would never be revealed.

3 Compiling adaptive executions

3.1 Shared continuation values

The first compiler pass computes the continuation values used by both training interfaces. Fix a sampled bank with K answer classes and total counts $\mathbf{n} = (n_1, \ldots, n_K)$. When the count state $\mathbf{c} = (c_1, \ldots, c_K)$ is sufficient, c_j records how many responses of class j have been revealed, and $t = \sum_j c_j$. Class identities remain attached to the coordinates so that terminal answer selection is well defined. Histories with the same counts share one continuation value. Conditional on revealing another candidate, the probability that it belongs to class j is

$$P_j(\mathbf{c} \mid B) = \frac{n_j - c_j}{N - t}, \qquad n_j > c_j. \tag{3}$$

The denominator is the remaining superpool size, including when the deployment horizon satisfies $H < N$.

Let $V(\mathbf{c})$ be expected terminal quality minus future acquisition costs and let $Q(\mathbf{c})$ be the value of continuing. Backward recursion gives

$$Q(\mathbf{c}) = \sum_{j : n_j > c_j} P_j(\mathbf{c} \mid B) \left[-\lambda_H + V(\mathbf{c} + \mathbf{e}_j) \right], \tag{4}$$

$$V(\mathbf{c}) = p_\phi(\mathbf{c}) u(\mathbf{c}) + [1 - p_\phi(\mathbf{c})] Q(\mathbf{c}), \quad t < H, \tag{5}$$
$$V(\mathbf{c}) = u(\mathbf{c}), \quad t = H.$$

The desired conditional return is $V(\mathbf{0})$; set $u(\mathbf{0}) = 0$ since stopping there is disallowed. Here $p_\phi(\mathbf{c})$ is the controller evaluated on its permitted observation. The bank totals in Equation (3) are compiler inputs and are never exposed to the controller. If the rule needs additional memory, use states $(\mathbf{c}, m)$ and retain its update on every transition; Appendix A.2 gives the corresponding recursion.

Theorem 1 (Exact compilation and superpool evaluation)**.** *For a fixed bank whose counts are sufficient, Equations* (3)*–*(5) *give the conditional expected deployment return at the root. The same holds for the memory-augmented recursion. Moreover, for $H \leq N$,*

$$V_{N \to H}(B) = \binom{N}{H}^{-1} \sum_{S \subseteq [N] : |S| = H} V_H(B_S), \tag{6}$$

where each $V_H(B_S)$ already averages reveal order and controller actions within the subset. Thus one recursion integrates subset formation and adaptive execution.

The proof is backward induction plus the fact that the first H positions of a uniform permutation form a uniform ordered subset. For $N = 16, H = 8$, explicit subset-and-order enumeration would require $\binom{16}{8} 8! = 518{,}918{,}400$ executions. The count-state recursion shares repeated continuation problems instead. Appendix A gives the proofs.

3.2 FORWARD OCCUPANCY AND ROLLOUT CREDITS

Generator training also needs to associate these values with the original responses. The forward pass tracks their participation in adaptive executions by propagating the probability $\alpha(\mathbf{c})$ of reaching each state before its stop/continue decision:

$$\alpha(\mathbf{0}) = 1, \qquad \alpha(\mathbf{c} + \mathbf{e}_j) \mathrel{+}= \alpha(\mathbf{c})[1 - p_\phi(\mathbf{c})]P_j(\mathbf{c} \mid B). \tag{7}$$

For candidate i in answer class $j = z_i$, define

$$q_i(B) = \frac{1}{n_j} \sum_{\mathbf{c}: t < H} \alpha(\mathbf{c})[1 - p_\phi(\mathbf{c})]P_j(\mathbf{c} \mid B) \left[-\lambda_H + V(\mathbf{c} + \mathbf{e}_j)\right], \tag{8}$$

$$\rho_i(B) = \frac{1}{n_j} \sum_{\mathbf{c}: t < H} \alpha(\mathbf{c})[1 - p_\phi(\mathbf{c})]P_j(\mathbf{c} \mid B). \tag{9}$$

Terms with $c_j = n_j$ vanish. The factor $1/n_j$ distributes class-level transition mass over the original exchangeable candidate identities. The coefficient q_i is a compiled return-to-go credit: it averages the return-to-go associated with revealing i, with zero contribution from paths that do not reveal it. It is neither a per-response correctness reward nor a centered advantage.

State compression preserves response identity at the gradient interface. Responses in the same answer class receive the same compiled coefficient, but their language-model score vectors can differ because their token sequences differ. The class computation reduces controller work without replacing individual response gradients. To see the factor $1/n_j$, conditional on reaching $\mathbf{c}$, candidate i in class j remains unrevealed with probability $(n_j - c_j)/n_j$. Multiplying by its subsequent selection probability $1/(N - t)$ gives $P_j(\mathbf{c} \mid B)/n_j$ (Appendix A.4).

The same forward pass returns reveal probabilities. Hence $\mathbb{E}[\tau \mid B] = \sum_i \rho_i$. If the controller does not inspect response length, expected revealed output-token cost is $\sum_i \rho_i L_i$ for generated lengths L_i. Fixed profile weights combine credits linearly, $q_i^{\mathcal{P}} = \sum_m w_m q_i^{(m)}$.

Complexity. Given the utility and stop probability at each of $|\mathcal{S}|$ reachable states, the backward and forward passes require $O(K|\mathcal{S}|)$ arithmetic operations, plus $O(N)$ to map class outputs to responses. For a pure count state,

$$|\mathcal{S}| \le \min \left\{ \prod_{j=1}^{K} (n_j + 1), \ \binom{H + K}{K} \right\}. \tag{10}$$

The state count is polynomial in H for fixed K but can be exponential when the number of answer classes grows with the bank. The gain comes from shared subproblems, not from a universal polynomial-time guarantee for arbitrary adaptive programs. Algorithm 2 appears in Appendix A.7.

Answer diversity determines how much history can be shared. Consider $N = 16, H = 8$, before pruning states excluded by forced early stopping. A bank with two answer classes of eight candidates each has only 45 feasible count states: pairs (c_1, c_2) satisfying $c_1 + c_2 \le 8$. If all 16 candidates instead have distinct answers, there are $\sum_{t=0}^{8} \binom{16}{t} = 39{,}203$ states. Both banks admit the same 518,918,400 ordered prefixes. Repeated answers therefore create substantial sharing, while diverse banks can remain expensive at the same candidate budget. Additional controller memory can prevent otherwise identical count states from merging. Values and occupancies require $O(|\mathcal{S}|)$ storage; caching transitions adds $O(K|\mathcal{S}|)$ storage.

4 TRAINING THE GENERATOR AND CONTROLLER

The compiled credits and values connect the adaptive deployment objective to two parameter updates (Algorithm 1). Generator parameters change the response distribution: compiled causal credits weight the score of each original response. Controller parameters change stop/continue probabilities on a fixed bank's state graph: their update differentiates the value recursion. Both integrate executions conditional on sampled banks.

Algorithm 1 One compiled bank, two optimization interfaces

Require: IID bank B, horizon $H \le N$, utility u, controller p_ϕ, cost λ_H
Group candidates by answer and enumerate sufficient states $\mathcal{S}$
Set $V(\mathbf{c}) = u(\mathbf{c})$ when $t = H$ or stopping is forced
for nonterminal states with $p_\phi(\mathbf{c}) < 1$ in decreasing revealed count **do**
    Compute continuation Q and value V using Equations (4)–(5)
end for
Set all occupancies and credit/reveal accumulators to zero; $\alpha(\mathbf{0}) \leftarrow 1$
for nonterminal states with $p_\phi(\mathbf{c}) < 1$ in increasing revealed count **do**
    Propagate occupancy and accumulate credits using Equations (7)–(9)
end for
Map class coefficients to the original candidates $i = 1, \dots, N$
Generator: weight sequence scores by q_i with the controller fixed
Controller: differentiate $V(\mathbf{0})$ with the bank fixed

4.1 EXACT-G: GENERATOR POLICY GRADIENT

Hold the controller fixed and write $\mathbf{h}_i = \nabla_\theta \log \pi_\theta(Y_i \mid x)$. The compiled credits from Equation (8) weight each original response's score:

$$G_{\mathrm{comp}}(B) = \sum_{i=1}^{N} q_i(B)\mathbf{h}_i. \tag{11}$$

EXACT-G samples response banks, combines compiled credits using the profile weights, and holds these credits fixed when weighting sequence log probabilities. For M prompt banks, the unscaled surrogate is

$$\mathcal{L}_{\mathrm{seq}} = -\frac{1}{M}\sum_{b=1}^{M}\sum_{i=1}^{N} \mathrm{sg}\big(q_{bi}^{\mathcal{P}}\big) \sum_{\ell=1}^{L_{bi}} \log \pi_\theta(y_{bi\ell} \mid x_b, y_{bi,<\ell}). \tag{12}$$

Here, sg stops differentiation through the credits, and L_{bi} is the token length of response i for prompt b. The loss gradient is the negative compiled estimator. Fixed global rescalings preserve the response weighting and gradient direction; per-response length normalization or per-batch credit normalization would define a different estimator. Appendix B.2 gives the training implementation.

To establish this update's meaning, consider the corresponding causal score-function estimator (Schulman et al., 2015). For one execution that reveals indices $\sigma_1, \dots, \sigma_\tau$,

$$G_{\mathrm{path}}(B, \xi) = \sum_{t=1}^{\tau} \left[u(s_\tau) - \lambda_H(\tau - t + 1)\right] \mathbf{h}_{\sigma_t}. \tag{13}$$

Only costs incurred at and after acquisition t enter that response's return-to-go. The following identities hold conditional on a prompt and then average over the prompt distribution.

Theorem 2 (Compiled generator policy gradient)**.** *Assume IID response generation and a nonanticipating controller with finite horizon. The extractor, controller, and utility have no direct dependence on θ, and differentiation can be interchanged with expectation. Then*

$$\mathbb{E}_B[G_{\mathrm{comp}}(B)] = \nabla_\theta J_H, \qquad G_{\mathrm{comp}}(B) = \mathbb{E}_\xi[G_{\mathrm{path}}(B, \xi) \mid B]. \tag{14}$$

When second moments exist,

$$\mathrm{Cov}(G_{\mathrm{path}}) - \mathrm{Cov}(G_{\mathrm{comp}}) = \mathbb{E}_B[\mathrm{Cov}(G_{\mathrm{path}} \mid B)] \succeq 0. \tag{15}$$

The statements extend by linearity to any fixed weighted deployment profile built from supported operators.

Rao–Blackwellization supplies the statistical identity (Liu et al., 2019); the count-state compiler makes its conditional expectation tractable for the adaptive program considered here. Equation (15) removes execution variance while retaining variation across prompts and sampled response banks.

Weighting the complete bank score $\sum_i \mathbf{h}_i$ by the single value $V(\mathbf{0})$ would also give an unbiased generator estimator. Our credits instead retain causal participation: at the execution level they exclude unrevealed responses and costs incurred before each acquisition, then average the remaining terms. The q_i coefficients therefore need not equal the bank value or sum to it. The covariance guarantee concerns the corresponding causal path estimator; it neither compares different objectives nor guarantees higher final utility after a finite training budget.

4.2 EXACT-C: CONTROLLER POLICY GRADIENT

Now hold the generator fixed and let the stop policy p_ϕ be differentiable. The count-state topology and class-transition probabilities do not depend on ϕ, so the same value recursion can be differentiated directly.

Proposition 3 (Exact controller policy gradient)**.** *For a fixed bank, with terminal utility and acquisition cost independent of ϕ,*

$$\nabla_\phi V(\mathbf{0}) = \sum_{\mathbf{c}:t<H} \alpha_\phi(\mathbf{c}) \nabla_\phi p_\phi(\mathbf{c}) \left[u(\mathbf{c}) - Q_\phi(\mathbf{c})\right]. \tag{16}$$

This equals the conditional expectation of the corresponding sampled controller-policy-gradient estimator over reveal order and stochastic stop/continue actions.

Equation (16) compares stopping now with continuing from each state, weighted by the probability of reaching that state. Optimizing its mean over frozen banks requires neither new language-model generation nor language-model backpropagation. A useful intermediate estimator samples K reveal orders but analytically averages stopping decisions along each ordered prefix chain; we call it *Order-MC-K*. Comparing full compilation with Order-MC isolates the effect of averaging over reveal order and subset selection, with stopping decisions already marginalized in both.

5 EXPERIMENTS

We evaluate execution error in generator-gradient space, deployment performance after generator training, and controller gradient precision and cost. The generator study compares EXACT-G with correctness GRPO; the controller study compares estimators of the same objective on frozen banks.

5.1 EXECUTION MARGINALIZATION IN GENERATOR-GRADIENT SPACE

We measure execution variation after response credits are mapped into generator parameter gradients. The diagnostic uses Qwen3-1.7B (Yang et al., 2025) and the tangent space of a rank-eight LoRA parameterization (Hu et al., 2022) on 84 mathematical prompts with 16 responses each. Full-support sequence scores at temperature 0.8 define a Gram matrix $K_{ij} = \mathbf{h}_i^\top \mathbf{h}_j$, which gives $\|G(q) - G(\tilde{q})\|^2 = (q-\tilde{q})^\top K (q-\tilde{q})$ without retaining every parameter-gradient vector. Direct weighted-backward checks match the Gram computation within numerical tolerance.

Table 1: **Execution noise remains substantial after mapping credits into real generator gradients.** Qwen3-1.7B, $H = 16$, 84 fixed prompts. The pooled noise share is the fraction of sampled-gradient second-moment energy attributable to execution variation. "Near exact" requires mean relative error ≤ 0.05 and mean cosine similarity ≥ 0.995, both averaged across estimator replicates for each prompt.

Controller	Pooled noise	Median noise	MC-32 near exact	MC-128 near exact
Aggressive	51.14%	70.54%	2.38%	8.33%
Medium	15.74%	24.08%	34.52%	52.38%

After averaging 128 sampled orders, only 8.33% and 52.38% of prompts meet the near-exact criterion under aggressive and medium stopping, respectively (Table 1). In the nested $16 \to 8$ diagnostic, subset variation contributes 31.69% of pooled execution-plus-subset gradient-error energy, rising to 53.70% in the predeclared stratum with at least five answer classes (Appendix B.4). Compilation therefore removes material execution error that finite Monte Carlo averaging leaves behind.

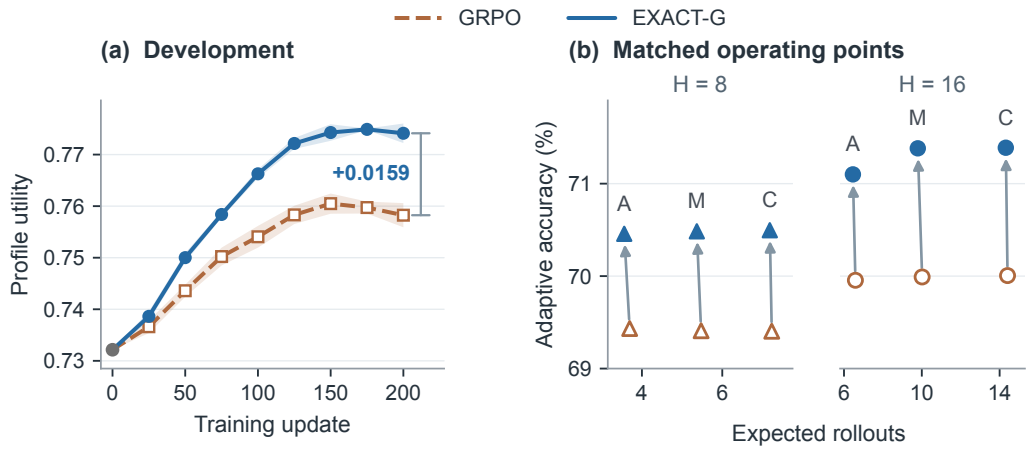

Figure 2: **EXACT-G improves development utility and shifts mean final operating points toward higher accuracy and fewer rollouts.** (a) Three-seed means at measured checkpoints, with ± one seed standard deviation; both curves share the initial-model anchor. (b) Each arrow connects operating points for the same frozen controller under GRPO and EXACT-G at development-selected checkpoints; horizons share the accuracy scale. A/M/C denote aggressive, medium, and conservative controllers.

Table 2: **EXACT-G improves deployment utility with higher mean adaptive accuracy and fewer rollouts.** Selected-checkpoint means over three paired training seeds and six equally weighted datasets. Accuracy columns are percentages; tokens count expected revealed output tokens per prompt. Adaptive metrics average the six fixed policies, with utility $\mathbb{E}[u - 0.2\tau/H]$. Seed uncertainty is reported in Table 13.

	Utility	Adapt. acc.	Rollouts	Pass@1	SC@8	SC@16	Tokens (k)
GRPO	0.5641	69.70	7.902	66.12	69.40	70.01	117.67
EXACT-G	**0.5780**	**70.88**	**7.780**	**66.63**	**70.50**	**71.39**	**116.28**
Difference	+0.0140	+1.19	−0.122	+0.52	+1.09	+1.38	−1.39

5.2 EXACT-G: TRAINING FOR DEPLOYMENT UTILITY

We post-train Qwen3-4B with EXACT-G for the six-term deployment profile in Equation (2), using compiled rollout credits in Equation (12). The comparison with correctness GRPO (Shao et al., 2024) evaluates the complete deployment-aware training method, without isolating compilation. Three paired seeds share the initial checkpoint and a POLARIS Stage-1-based recipe (An et al., 2025). Each arm uses 200 full-parameter updates with 64 prompts and 16 fresh IID responses per prompt: 204,800 training responses per seed. Prompt order and rollout seeds are paired across arms. Both use temperature 1.4, full-support sampling, and fixed calibration-based sequence-loss scales (Appendix B.2).

We evaluate a shared 512-prompt development set every 25 updates through update 200. Within each seed, each arm independently selects the checkpoint with the highest six-term development utility among these eight checkpoints. Final evaluation uses only the selected checkpoints, with the same decoding, answer extraction, fixed adaptive controllers, and $N = 16$ candidate-bank size across arms. Final aggregate metrics give equal weight to the six benchmark datasets.

Deployment utility rises by 0.013980, macro adaptive accuracy rises from 69.70% to 70.88%, and expected rollouts fall by 1.54% (Table 2). All three paired seeds improve these three metrics together. About 85% of the utility gain comes from correctness and 15% from reduced horizon-normalized acquisition cost.

Deployment utility improves under both uncertainty summaries: its paired gain has a three-seed 95% Student t interval of $[0.0029, 0.0251]$ and a paired-prompt interval of $[0.00134, 0.02631]$. Expected

rollouts also decrease under both. Adaptive accuracy has a positive seed interval, $[0.21, 2.16]$ percentage points, but its prompt interval $[-0.09, 2.43]$ includes zero. Seed variation on the fixed suite and prompt uncertainty are measured separately (Appendix C.3).

The development-utility difference grows from 0.00202 at update 25 to 0.01589 at update 200, plateauing late in training (Figure 2a). All six fixed policies yield higher mean macro accuracy with fewer expected rollouts after EXACT-G training (Figure 2b), as do all six dataset means. With controllers fixed, these shifts reflect changes in generated response banks.

SC@16 rises by 1.38 percentage points, compared with 0.52 points for Pass@1. The larger multi-response gain is consistent with deployment-aware training. Adaptive accuracy increases by 2.37 and 3.09 points on AIME24 and AIME25, compared with 0.23–0.91 points on the other four benchmarks (Table 10). Equal-dataset averaging captures these differing effect sizes. Token savings are less consistent: mean expected output-token cost falls by 1.18%, but increases in two of three seed pairs. This metric counts tokens consumed by the sequential policy, not parallel decoding latency; fewer acquisitions need not imply shorter responses or lower token cost.

5.3 EXACT-C: CONTROLLER GRADIENT PRECISION AND COST

We optimize a stop/continue policy over frozen Qwen3-4B responses to compare gradient precision, computation cost, and final utility. Training, development, and final evaluation use disjoint banks of 256, 512, and 1,590 prompts, each with 16 responses; final evaluation spans MATH500, Minerva Math, OlympiadBench, AMC23, AIME24, and AIME25. Trajectory policy gradient, Order-MC-1/16/64, and EXACT-C share the same small MLP actor and objective. A matched-environment PPO run adds a practical baseline (Schulman et al., 2017). Within each of the five training seeds, the primary estimators share actor initialization and prompt schedules; Appendix B.3 gives optimizer and selection details.

Table 3: **Controller optimization on frozen Qwen3-4B banks.** Final metrics at $H = 16$, $\beta = 0.2$, averaged equally across datasets and then across five training seeds; $\pm$ is seed standard deviation. CPU time covers controller training and scheduled development evaluation. All trained policies share the exact final evaluator.

Method	Accuracy (%)	Rollouts	Utility	CPU time (s)
Trajectory PG	67.26	3.138	0.6334 ± 0.0062	74.3 ± 5.6
PPO	67.38	3.095	0.6352 ± 0.0011	73.3 ± 1.2
Order-MC-1	67.37	2.992	0.6363 ± 0.0011	145.1 ± 13.6
Order-MC-16	67.22	2.707	0.6383 ± 0.0018	225.3 ± 41.3
Order-MC-64	67.40	2.972	0.6369 ± 0.0009	484.5 ± 8.7
EXACT-C	67.33	2.862	0.6376 ± 0.0015	389.8 ± 8.6

EXACT-C computes exact conditional controller gradients with a median per-bank computation time of 9.00 ms; Order-MC-64 takes 14.08 ms and retains 4.26% median relative error (Figure 3). Mean controller-training CPU time is 389.8 versus 484.5 seconds, a 19.5% reduction (Table 3). This timing includes scheduled development evaluation and checkpoint saving, but excludes LM generation, initial topology construction, learning-rate pilots, and final evaluation.

Held-out macro utility is 0.6376 for EXACT-C and 0.6369 for Order-MC-64. Their paired difference is 0.000701 with a joint seed/prompt bootstrap 95% interval of $[-0.000850, 0.003206]$, supporting comparable observed quality without establishing superiority or statistical equivalence.

Order-MC-16 has the highest mean endpoint utility; PPO and smaller MC budgets remain attractive at lower compute cost. Compilation benefits from a median of only 84 reachable states, but diverse

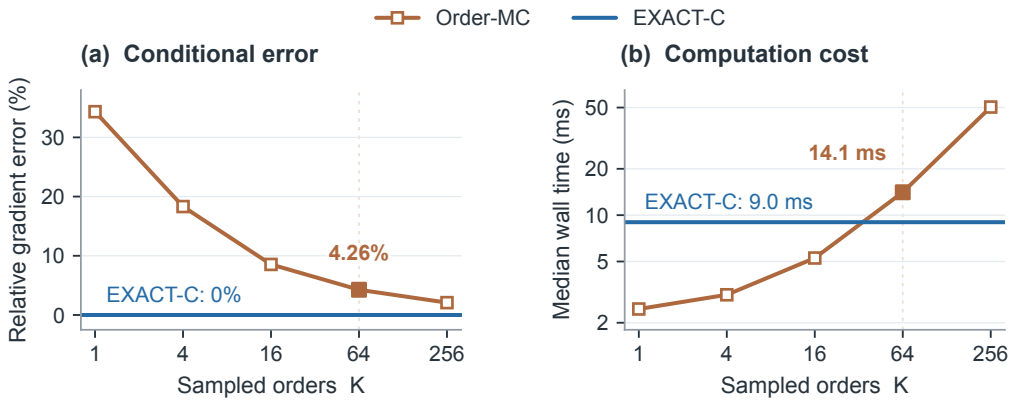

Figure 3: **Exact compilation removes conditional controller-gradient error at lower median cost than Order-MC-64 in this regime.** (a) Median relative gradient error across 256 frozen banks and estimator replicates versus the number K of sampled reveal orders. (b) Median per-bank gradient wall time on a single CPU thread. Exact compilation is independent of K and has zero conditional approximation error.

banks raise its 95th-percentile cost above MC-64 (Appendix C.2). Horizon and cost-preference checks likewise yield comparable mean utility with lower mean training CPU time for EXACT-C.

6 RELATED WORK

Training objectives tied to inference. Polychromic objectives optimize trajectory collections in reinforcement learning (Hamid et al., 2026); Poly-EPO optimizes sets of language-model responses (Orney et al., 2026). Our compiler evaluates the subset/order/stopping expectation of an adaptive deployment objective to compute causal response credits.

Adaptive inference operators. Adaptive-Consistency and Early-stopping Self-Consistency adapt acquisition to observed agreement (Aggarwal et al., 2023; Li et al., 2024). Closest to our controller application, Dai et al. (2026) learn a stopping/acquisition policy for a frozen language model, balancing correctness, sampling rounds, and sample count. Our compiler supplies generator credits as well as exact conditional controller gradients. The matched-environment PPO baseline does not reproduce their full action space or objective.

Analytic marginalization of stochastic computation. Stochastic computation graphs formalize score-function differentiation (Schulman et al., 2015); Rao–Blackwellization averages selected randomness (Liu et al., 2019). Knowledge compilation supports repeated inference (Darwiche and Marquis, 2002), and DeepProbLog differentiates compiled probabilistic inference (Manhaeve et al., 2018). We specialize these principles to adaptive acquisition through exchangeable count states, original-response causal credits, and a superpool identity for language-model policy gradients.

7 DISCUSSION AND CONCLUSION

EXACT shares execution states while preserving response-level causal credits, supporting generator and controller training for a specified adaptive utility. In Qwen3 mathematical reasoning, generator training improves mean held-out deployment utility; separate diagnostics establish material execution error and a controller-gradient precision–cost advantage over Order-MC-64 in the studied regime.

Scope. The sampling law must remain fixed during execution, and compiled states must preserve all controller information. This excludes prompt-changing refinement and tree search. Richer memory enlarges the graph; even count states can grow exponentially with answer diversity. Empirical evidence is limited to Qwen3 mathematical reasoning and modest candidate banks.

REPRODUCIBILITY STATEMENT

Appendix A gives the exactness and gradient proofs; Appendix B specifies the deployment profile, model and decoding settings, training, and checkpoint selection. The accompanying anonymous source archive includes aggregate results, frozen evaluation-prompt manifests, scripts for all quantitative figures and the generated result tables, and independent enumeration and finite-difference checks. Controller results use five training seeds and paired seed/prompt bootstrap uncertainty; generator results use three paired training seeds with Student t intervals (Appendix C.3).

AI USE STATEMENT

AI tools assisted with revising and formatting the manuscript, visualizing experimental results, and checking references and mathematical and numerical consistency. The authors retain responsibility for the final manuscript.

REFERENCES

Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let's sample step by step: Adaptive-consistency for efficient reasoning and coding with LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12375–12396, 2023. doi: 10.18653/v1/2023.emnlp-main.761. URL https://aclanthology.org/2023.emnlp-main.761/.

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. POLARIS: A post-training recipe for scaling reinforcement learning on advanced reasoning models. Technical report and open-source implementation, 2025. URL https://hkunlp.github.io/blog/2025/Polaris.

Runpeng Dai, Tong Zheng, Rui Liu, Chengsong Huang, and Hongtu Zhu. Small RL controller, large language model: RL-guided adaptive sampling for test-time scaling. *arXiv preprint arXiv:2606.03102*, 2026. URL https://arxiv.org/abs/2606.03102.

Adnan Darwiche and Pierre Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002. doi: 10.1613/jair.989. URL https://jair.org/index.php/jair/article/view/10311.

Jubayer Ibn Hamid, Ifdita Hasan Orney, Ellen Xu, Chelsea Finn, and Dorsa Sadigh. Polychromic objectives for reinforcement learning. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2509.25424.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen 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 *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, 2024. doi: 10.18653/v1/2024.acl-long.211. URL https://aclanthology.org/2024.acl-long.211/.

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 *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2106.09685.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2206.14858.

Yiwei Li, Peiwen Yuan, Shaoxiong Feng, Boyuan Pan, Xinglin Wang, Bin Sun, Heda Wang, and Kan Li. Escape sky-high cost: Early-stopping self-consistency for multi-step reasoning. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/3fe2a777282299ecb4f9e7ebb531f0ab-Abstract-Conference.html.

Runjing Liu, Jeffrey Regier, Nilesh Tripuraneni, Michael Jordan, and Jon McAuliffe. Rao-Blackwellized stochastic gradients for discrete distributions. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4023–4031, 2019. URL https://proceedings.mlr.press/v97/liu19c.html.

Robin Manhaeve, Sebastijan Dumančić, Angelika Kimmig, Thomas Demeester, and Luc De Raedt. DeepProbLog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://papers.nips.cc/paper/2018/hash/dc5d637ed5e62c36ecb73b654b05ba2a-Abstract.html.

Ifdita Hasan Orney, Jubayer Ibn Hamid, Shreya S. Ramanujam, Shirley Wu, Hengyuan Hu, Noah Goodman, Dorsa Sadigh, and Chelsea Finn. Poly-EPO: Training exploratory reasoning models. *arXiv preprint arXiv:2604.17654*, 2026. URL https://arxiv.org/abs/2604.17654.

John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, 2015. URL https://arxiv.org/abs/1506.05254.

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

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

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2203.11171.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

A Derivations and proofs

A.1 Conditions and notation

All statements can first be read conditional on a prompt x, then averaged over the prompt distribution. The bank consists of independent, identically distributed responses from $\pi_\theta(\cdot \mid x)$. The extractor, terminal utility, and controller are independent of θ when taking a generator gradient. The controller is nonanticipating: an acquisition decision depends on previously revealed responses and its own independent randomization, not on the next or unrevealed responses. The horizon is finite. We assume the score-function identity holds and the relevant derivatives are integrable; a fixed finite token horizon with differentiable full-support token probabilities is a sufficient setting. Covariance claims additionally require finite second moments.

The sufficient state must preserve both future decisions and terminal utility. For the count-only implementation, it includes the histogram with class identities, while the bank's total class counts are fixed compiler inputs. The actor may use a coarser observation of that state. If two histories have equal histograms but a controller distinguishes them through other memory, they cannot be merged unless that memory is included. Permutation symmetry of a bank supports the execution calculation; independence of model responses is additionally needed for the online generator-score argument below.

A.2 Retaining controller memory

If counts alone are insufficient, let m retain the controller's additional memory and define the compiler state $a = (\mathbf{c}, m)$. Revealing class j gives the successor $a_j^+ = (\mathbf{c} + \mathbf{e}_j, F(m, j, \mathbf{c}))$, where F is the fixed memory update. For example, a rule that breaks ties by first appearance must retain the ordering of class introductions. The remaining-class probabilities still depend only on $\mathbf{c}$, so

$$\begin{aligned} Q(a) &= \sum_{j:n_j>c_j} \frac{n_j - c_j}{N - t} \left[-\lambda_H + V(a_j^+)\right], \\ V(a) &= p_\phi(a)u(a) + [1 - p_\phi(a)]Q(a), \qquad t < H, \end{aligned} \tag{17}$$

with $V(a) = u(a)$ at $t = H$. In the occupancy, credit, and derivative equations, replace each count state by a and each successor by a_j^+. This preserves the proofs below because each acquisition still increases depth by one. The $1/n_j$ credit allocation remains valid whenever candidates within class j are indistinguishable to the controller and utility. If the controller observes additional candidate attributes, those attributes must be included in the class definition.

A.3 Proof of Theorem 1

Exact continuation value. At a state with $t = H$, the controller must terminate, so $V(\mathbf{c}) = u(\mathbf{c})$. Suppose the recursion is exact at all states at depth $t + 1$. After reaching $\mathbf{c}$ at depth t, stopping returns $u(\mathbf{c})$ with probability $p_\phi(\mathbf{c})$. Otherwise, a uniformly selected remaining candidate has class j with probability $(n_j - c_j)/(N - t)$, incurs cost λ_H, and enters $\mathbf{c} + \mathbf{e}_j$. The induction hypothesis evaluates the latter continuation exactly. Conditioning on this decision and transition yields Equations (4)–(5). Backward induction proves the root value is the conditional expected return.

Superpool identity. For a uniform permutation σ of $[N]$, every distinct ordered prefix $(i_1, \ldots, i_H)$ has probability $(N - H)!/N!$. Sampling a uniform unordered subset S of size H, followed by a uniform ordering of S, assigns probability

$$\binom{N}{H}^{-1} \frac{1}{H!} = \frac{(N - H)!}{N!} \tag{18}$$

to the same ordered prefix. A nonanticipating bounded controller has identical execution distributions under these constructions. Integrating the conditional return over the uniform subset establishes Equation (6). Early stopping does not affect the identity: the hypothetical remainder of the prefix can be sampled in advance and left unobserved. The transition denominator remains $N - t$ because every next acquisition is chosen from the remaining superpool, not from an explicitly constructed subset.

A.4 PER-CANDIDATE CREDIT UNDER COUNT-STATE COMPRESSION

Fix a candidate i in a class j containing n_j original candidates. Conditional on reaching a revealed count state $\mathbf{c}$, the probability that this particular candidate has not yet been revealed is $(n_j - c_j)/n_j$. This follows from exchangeability of identities within the class: swapping two original candidates in that class leaves every observed class sequence, controller decision, and terminal utility unchanged. If the controller continues, each of the $N - t$ remaining candidate identities is selected uniformly. Therefore the joint probability of reaching $\mathbf{c}$ and next revealing i is

$$\alpha(\mathbf{c})[1 - p_\phi(\mathbf{c})]\frac{n_j - c_j}{n_j(N-t)}. \tag{19}$$

The expected return-to-go associated with that reveal is $-\lambda_H + V(\mathbf{c} + \mathbf{e}_j)$. Summing their product over all pre-horizon states gives Equation (8). Summing just the probability gives Equation (9). A particular candidate is revealed at most once, hence ρ_i is its reveal probability and $\sum_i \rho_i = \mathbb{E}[\tau \mid B]$.

The $1/n_j$ allocation is essential. Dividing by the number of remaining candidates of class j would instead produce a coefficient conditional on a particular identity being unobserved, without accounting for the probability of that event. When identities carry attributes the controller actually uses, class membership must be refined or the state augmented; the exchangeability argument cannot then be applied using answer classes alone.

For $H < N$, let $q_i^H(B_S)$ be the subset compiler's credit, defined as zero for $i \notin S$. The same ordered-prefix argument proves

$$q_i^{N\to H}(B) = \binom{N}{H}^{-1} \sum_{S:|S|=H} q_i^H(B_S). \tag{20}$$

This identity concerns the original labeled candidate i, even though subset computations may have different class multiplicities.

A.5 PROOF OF THEOREM 2

Unbiased online score estimator. Consider the online execution before pool augmentation. The probability of a realized path factors into generator probabilities for the acquired responses and controller probabilities for its decisions. With the controller held fixed, differentiating the path probability gives the sum of acquired-response scores. Thus the likelihood-ratio identity gives

$$\nabla_\theta J_H = \mathbb{E}\left[R \sum_{t=1}^{\tau} \nabla_\theta \log \pi_\theta(Y_t \mid x)\right], \qquad R = u(s_\tau) - \lambda_H \tau. \tag{21}$$

Let I_t indicate that acquisition t occurs. I_t is measurable before Y_t is generated. Conditional on that history, an acquired response is a fresh draw from π_θ, so its score has mean zero. In particular,

$$\mathbb{E}\left[I_t \lambda_H (t-1) \nabla_\theta \log \pi_\theta(Y_t \mid x)\right] = 0. \tag{22}$$

Adding this zero-mean past-cost term to Equation (21) replaces R at acquisition t by $u(s_\tau) - \lambda_H(\tau - t+1)$. This yields the return-to-go estimator in Equation (13). An IID bank and a uniform permutation generate the same unconditional prefix law, so the augmented estimator has the same expectation.

Conditional credit identity. For a fixed bank, $\mathbf{h}_i$ is deterministic. The coefficient of $\mathbf{h}_i$ in a sampled execution is zero unless i is revealed, and otherwise equals its realized return-to-go. Appendix A.4 shows that its conditional expectation is exactly $q_i(B)$. Linearity therefore gives

$$\mathbb{E}_\xi[G_{\text{path}} \mid B] = \sum_i \mathbf{h}_i \mathbb{E}_\xi\left[\sum_{t=1}^{\tau} \mathbf{1}\{\sigma_t = i\}\{u(s_\tau) - \lambda_H(\tau - t + 1)\} \mid B\right] \tag{23}$$

$$= \sum_i q_i(B)\mathbf{h}_i = G_{\text{comp}}(B). \tag{24}$$

The tower property gives unbiasedness. Notice that the zero-mean argument for past costs uses the unconditional online IID law; it does not assume that without-replacement draws are independent conditional on a realized bank.

Covariance. The law of total covariance applied to G_{path} conditioned on B is

$$\mathrm{Cov}(G_{\mathrm{path}}) = \mathrm{Cov}(\mathbb{E}[G_{\mathrm{path}} \mid B]) + \mathbb{E}[\mathrm{Cov}(G_{\mathrm{path}} \mid B)]\,. \tag{25}$$

Substituting the conditional credit identity proves Equation (15). The second term is positive semidefinite, including after projection onto any fixed parameter direction. For L independent sampled executions conditional on the same bank, the residual conditional covariance of their average is $\mathrm{Cov}(G_{\mathrm{path}} \mid B)/L$. No claim follows that an optimizer with clipping, adaptive moments, or finite training time must have better final utility.

Profiles. For fixed weights w_m, take a weighted sum of path estimators for the profile terms. The executions can be sampled independently or coupled, provided each marginal execution law is correct. Its conditional expectation is $\sum_i (\sum_m w_m q_i^{(m)})\mathbf{h}_i$. Linearity of differentiation and expectation proves the profile gradient identity, and total covariance gives the corresponding dominance for that specified weighted estimator.

A.6 Proof of Proposition 3

Write $D(\mathbf{c}) = \nabla_\phi V(\mathbf{c})$. At forced terminal states $D = 0$, since u is fixed. Differentiating Equation (5) gives

$$D(\mathbf{c}) = \nabla_\phi p_\phi(\mathbf{c})[u(\mathbf{c}) - Q_\phi(\mathbf{c})] + [1 - p_\phi(\mathbf{c})] \sum_j P_j(\mathbf{c} \mid B) D(\mathbf{c} + \mathbf{e}_j). \tag{26}$$

Unrolling this acyclic recursion from the root assigns the local derivative at $\mathbf{c}$ exactly its reach probability $\alpha_\phi(\mathbf{c})$. This proves Equation (16). Forced actions have constant probabilities and zero derivatives. Alternatively, differentiate the finite sum over controller-action paths with the likelihood-ratio identity, omitting forced actions from the score sum; both calculations differentiate the same conditional expected value. This establishes the estimator identity without sampling the stopping path.

A.7 Algorithm and state complexity

Algorithm 2 Count-state value, credit, and controller-gradient compilation

Require: Bank classes $z_{1:N}$, horizon $H \le N$, utility u, stop policy p, cost λ_H

Count class multiplicities n_j and enumerate reachable sufficient count states $\mathcal{S}$
for states $\mathbf{c} \in \mathcal{S}$ in decreasing $t = \sum_j c_j$ **do**
  if $t = H$ or $p(\mathbf{c}) = 1$ **then**
    $V(\mathbf{c}) \leftarrow u(\mathbf{c})$
  else
    $P_j \leftarrow (n_j - c_j)/(N - t)$ for each available class j
    $Q(\mathbf{c}) \leftarrow \sum_j P_j[-\lambda_H + V(\mathbf{c} + \mathbf{e}_j)]$
    $V(\mathbf{c}) \leftarrow p(\mathbf{c})u(\mathbf{c}) + (1 - p(\mathbf{c}))Q(\mathbf{c})$
  end if
end for
Set all occupancies, class credits, and class reveal probabilities to zero; $\alpha(\mathbf{0}) \leftarrow 1$
for states $\mathbf{c} \in \mathcal{S}$ in increasing depth, with $t < H$ and $p(\mathbf{c}) < 1$ **do**
  for classes j with $c_j < n_j$ **do**
    $m \leftarrow \alpha(\mathbf{c})(1 - p(\mathbf{c}))(n_j - c_j)/(N - t)$
    $\alpha(\mathbf{c} + \mathbf{e}_j)$ += m
    q_j += $(m/n_j)[-\lambda_H + V(\mathbf{c} + \mathbf{e}_j)]$
    ρ_j += m/n_j
  end for
end for
Map class credits and reveal probabilities back to the N candidate identities
For learned p_ϕ, obtain $\nabla_\phi V(\mathbf{0})$ by differentiation or Equation (16)
return $V(\mathbf{0})$, $q_{1:N}$, $\rho_{1:N}$, and optional $\nabla_\phi V(\mathbf{0})$

At most $\prod_j (n_j + 1)$ count vectors satisfy the multiplicity bounds. Ignoring those bounds, the number of nonnegative K-dimensional count vectors with total at most H is $\binom{H+K}{K}$ by stars and bars. Taking the smaller upper bound proves Equation (10). Each state has at most K outgoing class transitions. Values and occupancies require $O(|\mathcal{S}|)$ storage, plus $O(N + K)$ for identities and class outputs if edges are generated when needed. Explicitly cached edges require $O(K|\mathcal{S}|)$ additional storage. A finite controller-memory component multiplies the reachable state count accordingly.

A.8 Token-cost evaluation and objective scaling

For fixed generated lengths L_i and indicators I_i of eventual reveal, total sequential output cost is $\sum_i I_i L_i$. Linearity yields

$$\mathbb{E}\left[\sum_i I_i L_i \mid B\right] = \sum_i L_i \mathbb{P}(I_i = 1 \mid B) = \sum_i L_i \rho_i(B). \tag{27}$$

This does not replace the per-rollout cost in the trained objective. It provides a second, token-based evaluation of the same policy. If stopping uses observed token lengths, identity symmetry within answer classes no longer suffices; the compiler must preserve the additional information. If all candidates were eagerly generated, their generation cost would already have been paid, so the expression describes sequential deployment rather than retroactive savings on bank construction.

Equation (12) uses a mean over prompts and a sum over candidates. Our implementation also divides by the fixed $N = 16$ and applies positive scalars fitted once on the initial calibration bank. These factors are constants with respect to the fresh training bank; they preserve response weighting and the gradient direction. In contrast, per-response token averaging or per-batch centering of compiled credits is a different estimator and is not used.

B Experimental definitions and implementation

B.1 Benchmarks, banks, and metrics

The main applications use the six frozen mathematical benchmark sets in Table 4. MATH500 is a 500-problem evaluation subset of MATH (Hendrycks et al., 2021). Minerva Math follows the quantitative-reasoning evaluation associated with Lewkowycz et al. (2022), and OlympiadBench uses the selected mathematical question set from He et al. (2024). Minerva Math, OlympiadBench, AMC23, and AIME24/25 use the frozen POLARIS evaluation inputs (An et al., 2025). These counts describe the evaluated manifests, not the sizes of every upstream benchmark release or variant. The anonymous source archive includes the development and final-evaluation manifests, with prompt identities, source indices, question text, and reference answers.

Table 4: **Frozen evaluation sizes.** Every prompt has a 16-response bank for each evaluated model. The development set is separate from the six final sets.

Set	Prompts	Responses per model
Development	512	8,192
MATH500	500	8,000
Minerva Math	272	4,352
OlympiadBench	675	10,800
AMC23	83	1,328
AIME24	30	480
AIME25	30	480
Final total	1,590	25,440

For prompt x with correctness flags r_{xi}, the Pass@1 estimate is $N^{-1} \sum_i r_{xi}$, followed by a mean over prompts. It estimates expected single-response correctness, not the probability of at least one correct response among 16. SC@16 applies terminal plurality to all 16 responses. SC@8 is the

exact conditional expectation of that decision over uniform eight-response subsets of the bank. It can therefore be fractional at the per-prompt level.

For a fixed adaptive policy m with horizon H_m, define per-prompt accuracy A_{xm}, expected acquisitions T_{xm}, and expected output-token cost C_{xm}. Its utility is $U_{xm} = A_{xm} - \beta T_{xm}/H_m$. For the six-term profile used in the main EXACT-G experiment, the per-prompt profile metrics are

$$\bar{A}_x = \frac{1}{6}\sum_m A_{xm}, \quad \bar{T}_x = \frac{1}{6}\sum_m T_{xm}, \quad \bar{C}_x = \frac{1}{6}\sum_m C_{xm}, \quad \bar{U}_x = \frac{1}{6}\sum_m \left(A_{xm} - \beta T_{xm}/H_m\right). \tag{28}$$

One must normalize the acquisition cost by its own horizon before averaging; dividing $\bar{T}_x$ by an average horizon is not equivalent. Dataset metrics average prompts, and the final macro averages the six dataset metrics equally. Reported differences are computed before rounding, so they may differ slightly from differences of displayed values. Controller-learning tables instead evaluate the learned policy at one (H, β), so their utility values are not directly comparable to the six-term generator-profile utility.

Answer semantics. The main Qwen3-4B evaluation extracts the last boxed answer, normalizes its string, and compares it against a schema-aware reference that can be a string, numeric answer, or list of accepted answers. Answer clustering uses the normalized prediction, not its correctness. Invalid responses consume an acquisition and their generated tokens but cast no vote; the leader fraction retains all acquired responses in its denominator. Invalid-only states cannot trigger an early stop, and no valid answer at the horizon is incorrect. Ties select the lexicographically smallest normalized valid answer. A class must have consistent correctness within a prompt.

Context length. The response cap is $\min(39936, 40960 - L_x^{\text{prompt}})$. The evaluation contract preserves the complete prompt rather than filtering or truncating it. For the two OlympiadBench prompts above the nominal 1,024-token prompt allowance, lengths 1,265 and 1,276 give caps 39,695 and 39,684. All other generation settings remain fixed across models. Generated-token cost counts response tokens only.

B.2 GENERATOR-TRAINING PROTOCOL

The initial model is `Qwen/Qwen3-4B` with thinking enabled. "Initial" in the result tables denotes this starting model, not the distinct `Qwen3-4B-Base` checkpoint. We use the POLARIS Qwen3-4B Stage-1 source recipe (An et al., 2025), with structural changes needed for the deployment-profile objective used by EXACT-G: 16 responses per prompt, no outcome-dependent resampling or group substitution, and a fixed-scale sequence-token-sum loss. This is not a reproduction of the final multistage POLARIS model.

Training data derive from the Stage-1 mathematical prompt pool. Normalized prompt matching removes final-test overlap and excludes the protected development and calibration prompts from active training. The normalization uses Unicode NFKC, collapsed whitespace, and removal of the fixed answer-instruction suffix for identity checking. Table 5 summarizes the shared settings. The prompt adds the POLARIS instruction to reason step by step and place the final answer in a box, once; it uses no few-shot examples or post-generation answer repair.

The paired seeds are 202609081, 202609082, and 202609083. In this order, development selection chooses updates $(150, 175, 150)$ for GRPO and $(175, 150, 200)$ for EXACT-G. Each trajectory runs on one node with four GH200 GPUs, full-shard data parallelism across all four GPUs, and rollout tensor-parallel size one. The 64 prompt groups form the global update batch, not a per-GPU batch.

The baseline uses the GRPO group advantage

$$A_i = \frac{r_i - \bar{r}}{s_r + 10^{-6}}, \qquad \bar{r} = \frac{1}{N}\sum_i r_i, \qquad s_r^2 = \frac{1}{N-1}\sum_i (r_i - \bar{r})^2. \tag{29}$$

The EXACT-G arm uses the uncentered, equal-weight average of the six compiled profile credits. Once the controllers are fixed, a shared collection $\mathcal{B}$ of initial calibration banks fixes

$$s_{\text{len}} = \left(\frac{1}{|\mathcal{B}|N}\sum_{B\in\mathcal{B},i} L_i\right)^{-1}, \qquad s_{\text{credit}} = \frac{\text{RMS}_{\mathcal{B},i}(A_i)}{\text{RMS}_{\mathcal{B},i}(q_i^{\mathcal{P}})}. \tag{30}$$

The baseline receives $s_{\text{len}} A_i$ and the EXACT-G arm receives $s_{\text{len}} s_{\text{credit}} q_i^{\mathcal{P}}$. Both are applied to response-token log-probability sums, with a mean over the fixed number of sequences. The scales stay fixed throughout training. The compiled credits are not batch-centered, divided by their current standard deviation, or normalized by each response length. Each fresh bank is used for one accumulated update and then discarded.

Table 5: **Matched Qwen3-4B generator settings.**

Setting	Value
Adaptation	Full-parameter training, bf16
Training updates	200 per arm per seed
Paired training seeds	3
Prompt groups / update	64
Responses / prompt	16 fresh IID responses
Training responses / arm / seed	204,800
Temperature; top-p; top-k	1.4; 1.0; disabled
Nominal response / context cap	39,936 / 40,960 tokens
Learning rate; weight decay	10^{-6}; 0.01
Global gradient-norm clip	1.0
KL penalty; entropy bonus	0; 0
PPO ratio clip (lower / upper)	0.20 / 0.28
Optimizer steps / fresh bank	1, after all microbatches
Development checkpoints	Every 25 updates, from 25 through 200
Selection	Maximum six-term development utility
Tie-break; initial-model eligibility	Earlier update; initial model not eligible

Experimental deployment profile. A rule stops when its minimum acquisition count, leader fraction threshold q, and leader–runner-up margin d are all met, or at the horizon. The frozen aggressive, medium, and conservative rules use $(m_{16}, q, d) = (6, 0, 2), (9, 5/8, 2), (14, 8/11, 2)$, respectively. The minimum is $m_H = \max(3, \lceil m_{16} H/16 \rceil)$, giving $3, 5, 7$ acquisitions at $H = 8$. Training and headline development and final evaluation use temperature 1.4, top-$p = 1$, and no top-k truncation.

The initial 256-prompt calibration bank chooses the triple using expected stopping fractions without consulting correctness. The search considers $m_{16} \in \{3, \ldots, 14\}$, thresholds $q = k/t$ for $3 \le t \le 16$ and $0 \le k \le t$, and margin $d = 2$. At each horizon, stopping fractions must have adjacent spacing at least 0.10, aggressive-to-conservative span at least 0.20, and conservative fraction at most 0.95. Selection maximizes the smaller span across horizons, then minimizes the medium fraction's deviation from the midpoint, then applies a lexicographic parameter tie-break. Calibration at $H = 8$ uses exact $16 \to 8$ integration. These policies and $\beta = 0.2$ remain fixed throughout generator training and evaluation.

B.3 CONTROLLER-TRAINING PROTOCOL

The controller experiment reuses the initial Qwen3-4B 256-prompt calibration bank for training, the 512-prompt bank for development, and the six final banks for evaluation. No prompt is split across these roles. This reuse changes no generator-training dataset roles: the controller experiment has a frozen generator and its own policy parameters.

At each state, the actor observes seven values: the five largest valid answer counts divided by H, zero-padded; t/H; and the entropy of the valid-answer frequency distribution divided by $\log H$. Its MLP has hidden widths $(32, 64, 64, 32)$, tanh nonlinearities, and one sigmoid stopping output. The initial output bias is -2. The action is stop or acquire one response. Acquisition is forced before $t = 2$ and whenever no valid answer exists before H; termination is forced at H. The observation excludes question identity, benchmark identity, answer text, correctness labels, unrevealed counts, and future response lengths.

Five primary estimators share this actor and the objective $u - 0.2\tau/16$. Trajectory PG samples an order and stop/continue actions and uses return-to-go with an action-independent baseline. Order-MC-K samples $K \in \{1, 16, 64\}$ independent orders and exactly evaluates the stopping expectation along

each prefix chain. EXACT-C sums both sources of execution randomness through the count-state program. All primary runs use 1,000 actor updates, 32 uniformly sampled training prompts per update, Adam without weight decay or entropy bonus, a discount factor of one, and gradient-norm clipping at five. Prompt schedules and initial actor weights are paired across arms within each seed.

Each primary arm uses the same development-only learning-rate pilot over $\{10^{-4}, 3 \times 10^{-4}, 10^{-3}\}$, with pilot seed 101 and 500 updates per candidate; all select 10^{-3}. The main runs then restart for seeds 0–4, varying actor initialization, training prompt schedules, and execution randomness while reusing the frozen response banks. Exact development evaluation is applied to every primary method at updates 0, 10, 25, 50, 100, 250, 500, 750, and 1,000. Each seed selects maximum development utility, breaking ties by earlier update.

The PPO control uses the same actor observations, actions, and true-correctness reward, with 200,000 decision timesteps per seed for seeds 0–4. It uses rollout batches of 2,048 decisions, four optimization epochs, minibatches of 256 decisions, discount factor 1, GAE parameter 0.95, ratio clipping at 0.2, and normalized advantages. Its critic has the actor's hidden widths with a scalar value output. Separate Adam optimizers train the actor and critic, with value-loss coefficient 0.5, no value clipping or entropy bonus, and gradient-norm clipping at five. A development-only pilot with seed 101 and 100,000 decisions per candidate selects learning rate 10^{-3} from the same three candidates. Exact development evaluation occurs initially and after crossing 10,000, 25,000, 50,000, 100,000, and 200,000 decisions; selection maximizes development utility with an earlier-checkpoint tie-break. PPO uses a clipped surrogate and critic, so its update count is not equated to that of the estimator comparison. It is a matched-environment adaptation, not a reproduction of Dai et al. (2026).

Controller runs use one PyTorch intra-operation thread and one inter-operation thread. Reported training CPU time is process CPU time measured with `time.process_time()`, covering the controller training loop, scheduled development evaluation, and checkpoint saving. It excludes response generation, initial count-state topology construction, learning-rate pilots, and final evaluation. The offline per-bank gradient timings in Figure 3 are wall-clock measurements from `time.perf_counter()`, covering objective evaluation and backpropagation. They exclude state-graph construction; Order-MC reveal orders are also drawn before timing. These per-bank measurements are distinct from end-to-end training time.

B.4 GRADIENT DIAGNOSTICS AND NESTED DECOMPOSITION

The model-gradient diagnostic uses all 84 prompts whose earlier 16-response banks had empirical correctness strictly between 0.1 and 0.9 and at least two distinct answer classes. Prompt identities were fixed before generating the new full-support bank, so the new outcomes could not change prompt membership. This is a mechanism study, not an independent final-test sample. The LoRA adapter uses rank eight, scaling 16, and zero dropout, and targets the attention projections and feed-forward gate/up/down projections. The base model is frozen, and no optimization is performed. Scores use the same $T = 0.8$ full-vocabulary policy as generation, with deterministic math-SDPA backward and FP32 weighted gradient accumulation.

For a fixed controller, the compiled-gradient energy of prompt x is $S_x = \|G_{\mathrm{comp}}(B_x)\|^2$. Its conditional execution error is

$$E_x = \mathbb{E}_\xi \left\|G_{\mathrm{path}}(B_x, \xi) - G_{\mathrm{comp}}(B_x)\right\|^2 . \tag{31}$$

The pooled execution-noise share is $\sum_x E_x / \sum_x (S_x + E_x)$. Its per-prompt version is $E_x/(S_x + E_x)$; hence the pooled share need not equal the mean or median per-prompt share. The error expectation is estimated from sampled executions, with the compiled gradient as the exact conditional reference. MC near-exact fractions use per-prompt metrics averaged over estimator replicates and require both a relative gradient error of at most 0.05 and cosine similarity at least 0.995.

For the nested experiment, G_0 averages two sampled executions on a random complementary $8 + 8$ partition of a 16-response bank. G_1 uses exact execution within the same two halves, with zero-padded candidate coordinates; G_2 exactly averages all size-eight subsets. Define $E_{\mathrm{exec}} = \mathbb{E}\left\|G_0 - G_1\right\|^2$ and $E_{\mathrm{subset}} = \mathbb{E}\left\|G_1 - G_2\right\|^2$. The subset share is the corresponding pooled fraction $E_{\mathrm{subset}}/(E_{\mathrm{exec}} + E_{\mathrm{subset}})$. The decomposition also records the finite-sample cross term $2\mathbb{E}\langle G_0 - G_1, G_1 - G_2\rangle$, rather than silently assuming its empirical estimate is zero. The all-prompt, at-least-five-classes, and below-0.5-concentration strata were fixed before analyzing this bank.

B.5 NUMERICAL VALIDATION

Tiny-bank enumeration independently checks the compiler over complete permutations and feasible stopping times. Cases cover all-correct, all-wrong, multiple classes, ties, invalid-only, and mixed-invalid banks, with near-boundary and interior stopping probabilities. The controller derivative is compared with an independent occupancy implementation and finite differences. Table 6 reports maximum discrepancies across these tests. Observation-isolation checks additionally verify that equal revealed observations produce equal actor probabilities even when hidden bank totals or correctness differ.

Table 6: **Independent compiler checks.** Maximum absolute discrepancies in float64; these are implementation checks, not downstream effect sizes.

Check	Maximum absolute discrepancy
Value versus enumeration	1.11×10^{-16}
Accuracy/acquisition/token metrics versus enumeration	1.78×10^{-15}
Controller gradient versus permutation enumeration	4.16×10^{-17}
Controller gradient versus occupancy derivative	1.73×10^{-17}
Controller gradient versus finite differences	1.38×10^{-11}

The 256 controller-training banks have a median of 84 reachable count states, a 95th percentile of 4,224, and a maximum of 27,648. Corresponding edge counts are 176, 22,560, and 193,536. These measurements explain why full compilation is practical here while leaving substantial variation in per-bank cost. Preprocessing and tail timings are retained separately from median gradient time in Table 11.

C ADDITIONAL RESULTS

C.1 FULL QWEN3-4B CAPABILITY AND DEVELOPMENT RESULTS

Table 7 reports the initial model and three-seed means for both development-selected training arms on every final benchmark. Checkpoints are selected separately within each arm and seed from the 200-update runs (Appendix B.2). Table 8 shows all eight measured development checkpoints; Table 9 reports expected output-token costs.

Table 7: **Final static capability (%).** P1 is mean single-response correctness, S8 is subset-averaged SC@8, and S16 is SC@16. Initial denotes Qwen3-4B before the matched post-training runs.

	Initial			GRPO			EXACT-G		
Dataset	P1	S8	S16	P1	S8	S16	P1	S8	S16
MATH500	90.72	91.84	92.00	91.85	92.94	93.13	91.95	93.13	93.40
Minerva	37.06	38.48	38.97	37.60	39.16	39.58	37.74	39.56	39.71
OlympiadBench	44.14	45.29	45.48	46.35	47.44	48.00	47.22	48.38	48.84
AMC23	91.19	94.55	95.18	92.50	95.43	95.98	92.55	95.60	96.39
AIME24	53.54	58.50	56.67	57.01	62.31	62.22	57.15	64.13	65.56
AIME25	67.92	75.08	76.67	71.39	79.15	81.11	73.19	82.18	84.44
Macro	64.10	67.29	67.49	66.12	69.40	70.01	66.63	70.50	71.39

Table 8: **All measured development checkpoints.** Three-seed means; accuracy is in percent and utility includes seed standard deviation. The shared initial model has adaptive accuracy 85.90% and utility 0.732151. Development utility selects checkpoints; it is not a final-test result.

	Adaptive accuracy		Utility ± seed SD		
Update	GRPO	EXACT-G	GRPO	EXACT-G	Δ utility
25	86.34	86.53	0.7366 ± 0.0012	0.7386 ± 0.0009	$+0.0020$
50	87.04	87.63	0.7436 ± 0.0012	0.7500 ± 0.0009	$+0.0064$
75	87.70	88.44	0.7502 ± 0.0017	0.7584 ± 0.0006	$+0.0082$
100	88.08	89.19	0.7541 ± 0.0021	0.7663 ± 0.0009	$+0.0122$
125	88.49	89.77	0.7583 ± 0.0018	0.7722 ± 0.0011	$+0.0139$
150	88.70	89.98	0.7605 ± 0.0020	0.7743 ± 0.0016	$+0.0138$
175	88.62	90.03	0.7597 ± 0.0012	0.7749 ± 0.0002	$+0.0152$
200	88.49	89.95	0.7582 ± 0.0023	0.7741 ± 0.0019	$+0.0159$

Table 9: **Expected adaptive output-token cost.** Three-seed means over each dataset's prompts and the same six profile policies. Token cost decreases in five of six dataset means, but increases on MATH500; it is not the acquisition-count penalty optimized during training.

Dataset	GRPO	EXACT-G	Difference
MATH500	48,874.444	48,944.471	$+70.027$
Minerva Math	74,087.420	72,506.722	$-1{,}580.698$
OlympiadBench	114,846.811	112,567.872	$-2{,}278.939$
AMC23	99,464.454	98,360.911	$-1{,}103.544$
AIME24	164,172.031	164,081.636	-90.395
AIME25	204,549.100	201,210.807	$-3{,}338.292$
Macro	117,665.710	116,278.737	$-1{,}386.973$

Table 10: **Favorable mean accuracy–compute shifts across all six final benchmarks.** Accuracy (%), expected rollouts, and utility average the same six fixed policies within each dataset.

	Accuracy ↑		Rollouts ↓		Utility ↑	
Dataset	GRPO	EXACT-G	GRPO	EXACT-G	GRPO	EXACT-G
MATH500	93.02	93.25	7.450	7.421	0.8052	0.8080
Minerva	39.37	39.67	8.363	8.168	0.2529	0.2592
OlympiadBench	47.70	48.61	7.865	7.730	0.3447	0.3561
AMC23	95.67	95.90	7.606	7.557	0.8290	0.8321
AIME24	62.32	64.69	7.865	7.749	0.4907	0.5164
AIME25	80.11	83.19	8.261	8.054	0.6619	0.6963

C.2 CONTROLLER GRADIENT PRECISION AND COST PREFERENCES

Table 11 includes every sampled order budget, using EXACT-C as the numerical reference. EXACT-C's median runtime advantage over MC-64 does not extend to the 95th percentile: high-diversity banks increase count-state cost. Table 12 reports all secondary (H, β) settings, without selection on final-test performance.

Training-seed standard deviations describe variation across five learned controllers on the same frozen banks. For the primary EXACT-C-minus-MC-64 utility comparison, the reported uncertainty additionally uses 2,000 paired bootstrap replicates over training seeds and prompts within datasets. Repeated candidate permutations are not treated as independent test questions.

Table 11: **Conditional controller-gradient error and computation cost.** The comparison uses 256 frozen training banks; relative errors are dimensionless. MC uses 32 estimator replicates per bank, trajectory PG 256. Times are wall-clock measurements on one CPU thread. Median time to construct the shared state graph is 0.884 ms; peak process memory is 609.65 MiB.

Estimator	Median rel. error	95th rel. error	Median ms	95th ms
Trajectory PG	6.2943	45.5945	2.08	4.71
Order-MC-1	0.3434	3.3510	2.46	2.83
Order-MC-4	0.1832	2.1151	3.04	3.52
Order-MC-16	0.0854	0.9521	5.27	6.47
Order-MC-64	0.0426	0.4760	14.08	18.13
Order-MC-256	0.0209	0.2442	50.38	64.34
EXACT-C	0	0	9.00	71.17

Table 12: **Controller horizon and cost-preference checks.** Final macro averages over five training seeds. The primary $H = 16, \beta = 0.2$ results are in Table 3. Utilities from different β/H objectives should not be ranked as if they were one metric.

Method	H	β	Acc. (%)	Rollouts	Utility	CPU (s)
EXACT-C	8	0.2	67.03	2.414	0.609935	258.5
Order-MC-64	8	0.2	67.03	2.414	0.609935	304.1
EXACT-C	16	0.1	67.42	3.026	0.655286	407.2
Order-MC-64	16	0.1	67.40	3.000	0.655287	573.7
EXACT-C	16	0.4	67.07	2.436	0.609813	471.7
Order-MC-64	16	0.4	67.07	2.435	0.609841	572.4

C.3 UNCERTAINTY ACROSS GENERATOR TRAINING SEEDS

Table 13 summarizes three paired training seeds. For differences d_1, d_2, d_3, the 95% interval is $\bar{d} \pm t_{0.975,2} s_d/\sqrt{3}$, with $t_{0.975,2} = 4.30265$. It measures seed variation on the fixed benchmark suite, without prompt resampling. Seed intervals favor EXACT-G for SC@8, adaptive accuracy, rollout cost, and utility; those for Pass@1, SC@16, and token cost include zero. Three pairs provide small-sample estimates, not guarantees across training conditions.

Separate paired-prompt 95% intervals are $[-0.09, 2.43]$ percentage points for adaptive accuracy, $[-0.169, -0.080]$ for expected rollouts, and $[0.00134, 0.02631]$ for utility. Utility increases and expected acquisitions decrease under both uncertainty summaries; the paired-prompt interval for adaptive accuracy includes zero. These intervals measure distinct sources of uncertainty; neither is a joint seed/prompt interval.

Table 13: **Three-seed generator uncertainty.** Means ± seed standard deviations; differences are paired within seed. Accuracy and its differences use percent and percentage points, respectively. Output-token quantities are in thousands. The last column is the paired 95% Student t interval.

Metric	GRPO	EXACT-G	Difference	95% interval
Deployment utility	0.5641 ± 0.0044	0.5780 ± 0.0017	$+0.0140 \pm 0.0045$	$[+0.0029, +0.0251]$
Adaptive accuracy (%)	69.70 ± 0.40	70.88 ± 0.21	$+1.19 \pm 0.39$	$[+0.21, +2.16]$
Expected rollouts	7.902 ± 0.027	7.780 ± 0.022	-0.122 ± 0.039	$[-0.218, -0.025]$
Pass@1 (%)	66.12 ± 0.34	66.63 ± 0.20	$+0.52 \pm 0.34$	$[-0.33, +1.36]$
SC@8 (%)	69.40 ± 0.42	70.50 ± 0.19	$+1.09 \pm 0.34$	$[+0.25, +1.93]$
SC@16 (%)	70.01 ± 0.45	71.39 ± 0.26	$+1.38 \pm 0.58$	$[-0.05, +2.82]$
Expected tokens (k)	117.67 ± 1.47	116.28 ± 6.63	-1.39 ± 7.75	$[-20.63, +17.86]$

C.4 SUPPORTING QWEN3-1.7B GENERATOR STUDY

The supporting study uses Qwen3-1.7B with rank-eight LoRA, thinking disabled, temperature 0.8, full-support sampling, and a 4,096-token response cap. The baseline is correctness *leave-one-out* policy gradient, $a_i = r_i - \frac{1}{N-1}\sum_{j \neq i} r_j$, not GRPO normalization. Both arms use 32 prompts with 16 responses each per update, a 2,048-prompt training pool, and 96 cumulative updates. Evaluation averages terminal ties uniformly, unlike the main 4B protocol's fixed lexicographic rule.

A separate 512-prompt development measurement evaluates updates 48, 80, and 96 with 16 responses per prompt (Table 14). At update 80, EXACT-G increases utility by 0.015470 and SC@16 by 1.79 percentage points; the utility gains at updates 48 and 96 are smaller. This supports generator learning with compiled credits but does not select or quantify the primary 4B result.

Table 14: **Supporting generator learning on Qwen3-1.7B.** Fresh 512-prompt development measurement; accuracies in percent. The intervals are descriptive paired-prompt 95% bootstrap intervals for profile-minus-correctness utility, not a new held-out checkpoint-selection test.

Method	Update	Utility	Pass@1	SC@8	SC@16
Correctness LOO	48	0.661361	71.04	77.61	78.65
EXACT-G	48	0.662756	70.56	77.55	79.33
Correctness LOO	80	0.665368	71.09	77.91	79.20
EXACT-G	80	0.680838	71.70	79.14	80.99
Correctness LOO	96	0.669693	72.22	78.49	79.07
EXACT-G	96	0.671012	72.19	78.30	79.47
Utility difference, u48		$+0.001394$	$[-0.011981, +0.014769]$		
Utility difference, u80		$+0.015470$	$[+0.001144, +0.029795]$		
Utility difference, u96		$+0.001320$	$[-0.015800, +0.018439]$		