ROLE-ADAPTIVE POLICY OPTIMIZATION FOR OFFLINE REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Policy regularization in offline reinforcement learning balances policy improvement against reliance on uncertain value estimates. This balance can differ between selecting actions for execution and supplying actions for critic bootstrapping, yet methods such as TD3+BC couple these roles through a shared policy. We propose Role-Adaptive Policy Optimization (RAPO), which adapts policy-update coefficients according to their roles in value learning and execution. RAPO learns these coefficients by differentiating through candidate policy updates formed using the base algorithm's actor objective. For TD3+BC, RAPO separates bootstrap and execution actors and adapts their coefficients independently: the bootstrap objective penalizes policy-induced changes in target values, while the execution objective evaluates a local policy-improvement surrogate. For IQL, whose value learning is already independent of the execution actor, RAPO preserves the original value updates and adapts only the inverse temperature in advantage-weighted policy extraction. Experiments on D4RL locomotion and AntMaze tasks show improvements over both base algorithms, with larger gains for TD3+BC, whose RAPO instantiation outperforms baselines on average.

1 INTRODUCTION

Many offline reinforcement learning (RL) methods improve a policy by balancing critic-guided updates against a reference derived from recorded behavior (Fujimoto & Gu, 2021; Tarasov et al., 2023). The critic is learned from a fixed dataset, so its estimates can be unreliable for actions poorly covered by that dataset (Levine et al., 2020). The regularization coefficient controls how strongly an update follows these estimates. A setting that is useful at one stage of training may become overly restrictive or permissive as the actor and critic evolve.

The appropriate balance also depends on the role of the policy. In TD3+BC, one actor selects actions for execution and, through its target copy, supplies next-state actions for critic bootstrapping. An update toward higher estimated values can improve the execution policy while changing the critic's regression targets. Conversely, limiting target movement can restrict useful execution updates. These two uses therefore need not favor the same regularization, even when they share a critic.

IQL learns values without using its extracted actor, and methods with separate target and evaluation policies allow different constraint strengths (Kostrikov et al., 2022; Xu et al., 2024). Policy extraction can substantially affect performance for a given value function (Hansen-Estruch et al., 2023; Park et al., 2024). Adaptive-constraint methods can learn regularization coefficients through differentiable actor updates (Tan et al., 2026). The evaluation criterion for such an update should reflect whether the policy supplies execution actions or critic targets.

We propose *Role-Adaptive Policy Optimization* (RAPO), which learns regularization coefficients by evaluating candidate policy updates according to how their actions are used. Execution updates are assessed for local policy improvement, while bootstrap updates are assessed for the stability of the critic targets they produce. RAPO differentiates through a candidate update to choose its coefficient and retains the base algorithm's actor objective for the actual update.

We instantiate this procedure in TD3+BC and IQL. TD3+RAPO separates execution and bootstrap actors and learns their coefficients independently. IQL learns Q and V from dataset actions without using its execution actor in either update (Kostrikov et al., 2022). IQL+RAPO therefore adapts only

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.

the inverse temperature of advantage-weighted policy extraction. An explicit bootstrap actor is used when the base method obtains critic targets from policy actions.

The contribution is a role-specific criterion for learning each policy-update coefficient while retaining the base actor losses. Across fifteen D4RL tasks, both instantiations improve the domain averages of their base algorithms, and TD3+RAPO has the highest overall average among the compared methods. Matched TD3 controls examine the execution score and bootstrap adaptation; a fixed-coefficient control tests whether learned endpoint values reproduce adaptation during training.

2 Background

2.1 Offline Learning Setup

The training experience in offline RL is a fixed dataset $\mathcal{D} = \{(s, a, r, s')\}$ collected by behavior policies (Levine et al., 2020). Each tuple records a state $s \in \mathcal{S}$, action $a \in \mathcal{A}$, reward r, and successor state s'. Learning uses these records without further environment interaction.

We seek a policy π that maximizes $J(\pi) = \mathbb{E}_{\rho_0, P, \pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$, where ρ_0 is the initial-state distribution, P is the environment transition kernel, r is the reward function, and $\gamma \in [0, 1)$ is the discount factor. The actor-critic methods below represent the policy by π_θ and estimate action and state values with Q_ϕ and V_ψ, respectively.

2.2 Base Actor-Critic Updates

TD3+BC and IQL differ in their actor objectives and in the actions used for critic bootstrapping. Let $\bar{Q}_j$ denote target critic j and $\bar{Q}_{\min} = \min_{j=1,2} \bar{Q}_j$ their pointwise minimum. Bars consistently mark target critics. Let sg denote the stop-gradient operator. We write the losses for nonterminal transitions; implementation details are given in Appendix B.2.

TD3+BC. TD3+BC combines Q maximization with behavior cloning (Fujimoto & Gu, 2021). In the loss convention used by RAPO, its actor minimizes

$$\mathcal{L}_\pi^{\mathrm{TD3+BC}}(\theta; \alpha) = \mathbb{E}_{(s,a)\sim\mathcal{D}} \left[-\frac{Q_{\phi_1}(s, \pi_\theta(s))}{S} + \frac{\|\pi_\theta(s) - a\|_2^2}{\alpha d_a} \right], \tag{1}$$

where $\alpha > 0$, d_a is the action dimension, and $S = \mathrm{sg}(\max\{\mathbb{E}_{s\sim\mathcal{D}}|Q_{\phi_1}(s, \pi_\theta(s))|, \epsilon\})$ for a small $\epsilon > 0$. Increasing α weakens behavior regularization relative to Q maximization. The critic loss is

$$\mathcal{L}_Q^{\mathrm{TD3}}(\phi) = \sum_{j=1}^{2} \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}} \left[\left(Q_{\phi_j}(s, a) - \mathrm{sg}\left(r + \gamma \bar{Q}_{\min}(s', \bar{\pi}_\theta(s')) \right) \right)^2 \right]. \tag{2}$$

In TD3+BC, the actor in Equation (1) selects execution actions; its target copy in Equation (2) selects the next-state actions used for critic bootstrapping. RAPO assigns these two uses to separate actors.

IQL. IQL learns values from dataset actions independently of its actor (Kostrikov et al., 2022):

$$\mathcal{L}_V(\psi) = \mathbb{E}_{(s,a)\sim\mathcal{D}} \ell_\tau\left(\mathrm{sg}(\bar{Q}_{\min}(s, a)) - V_\psi(s) \right), \tag{3}$$

$$\mathcal{L}_Q^{\mathrm{IQL}}(\phi) = \frac{1}{2} \sum_{j=1}^{2} \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}} \left[\left(Q_{\phi_j}(s, a) - \mathrm{sg}(r + \gamma V_\psi(s')) \right)^2 \right], \tag{4}$$

where $\ell_\tau(u) = |\tau - \mathbf{1}\{u < 0\}|u^2$ and $\tau \in [1/2, 1)$ select an upper expectile (the mean at $\tau = 1/2$). Given the detached advantage $A(s, a) = \mathrm{sg}(\bar{Q}_{\min}(s, a) - V_\psi(s))$, the actor minimizes

$$\mathcal{L}_\pi^{\mathrm{IQL}}(\theta; \beta) = -\mathbb{E}_{(s,a)\sim\mathcal{D}} \left[\exp(\beta A(s, a)) \log \pi_\theta(a \mid s) \right]. \tag{5}$$

The inverse temperature $\beta > 0$ controls how strongly policy extraction favors higher-advantage actions.

2.3 META-GRADIENT LEARNING

Meta-gradient learning trains a coefficient by differentiating through the update it controls (Xu et al., 2018). For an actor loss $\mathcal{L}_{\text{inner}}$ and dataset minibatch $\mathcal{B}_{\text{in}}$, an *inner update* changes θ with α fixed. A gradient step with learning rate η gives

$$\widetilde{\theta}(\alpha) = \theta - \eta \nabla_\theta \mathcal{L}_{\text{inner}}(\theta; \alpha, \mathcal{B}_{\text{in}}).$$

An *outer loss* $\mathcal{L}_{\text{outer}}$ scores the updated actor on another minibatch $\mathcal{B}_{\text{out}}$. Holding the starting parameters θ fixed, the chain rule gives

$$\frac{\mathrm{d}\mathcal{L}_{\text{outer}}(\widetilde{\theta}(\alpha); \mathcal{B}_{\text{out}})}{\mathrm{d}\alpha} = \left(\frac{\partial \widetilde{\theta}}{\partial \alpha}\right)^\top \nabla_{\widetilde{\theta}} \mathcal{L}_{\text{outer}}. \tag{6}$$

Gradient descent on the outer loss trains α: the inner loss trains the policy, and the outer loss trains its update coefficient.

3 METHOD

RAPO applies the meta-gradient procedure in Section 2.3 with outer losses tailored to execution and critic bootstrapping (Figure 1).

3.1 POLICY ROLES AND ADAPTIVE COEFFICIENTS

The execution policy π_E selects actions in the environment. When value learning requires a policy to supply next-state actions, RAPO assigns that task to a separate bootstrap policy π_B. For each active role i, write c_i for its method-specific positive actor-loss coefficient:

Action use	Actor	Coefficient	Outer criterion
Execution	$\pi_E = \pi_{\theta_E}$	c_E	Local value improvement
Critic-target actions	$\pi_B = \pi_{\theta_B}$	c_B	Target stability

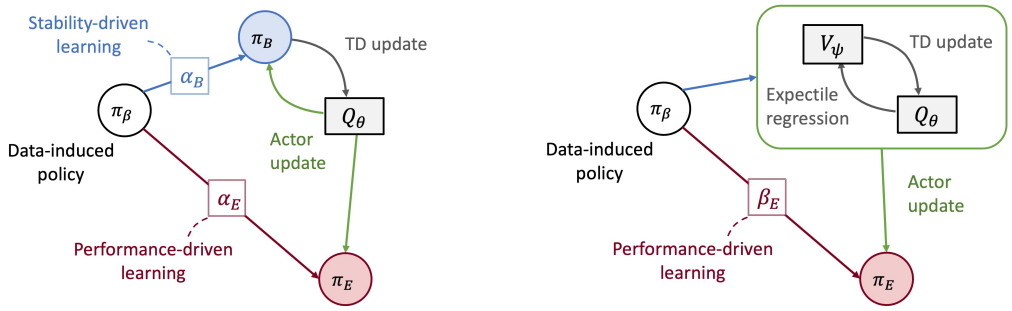
TD3+RAPO uses both actors with shared critics, with $c_E = \alpha_E$ and $c_B = \alpha_B$: π_E is executed, while π_B supplies the actions in Equation (2). IQL's target in Equation (4) is actor-independent, so its only required role is execution, with $c_E = \beta_E$.

For each active role i, the inner loss $\mathcal{L}_{\text{inner},i}$ is the corresponding actor objective from Section 2.2: Equation (1) with $\alpha = c_i$ for TD3+BC, or Equation (5) with $\beta = c_E$ for IQL.

Let $\mathcal{U}_i(\theta_i; c_i, \mathcal{B}_{\text{in}})$ denote one differentiable optimizer step on $\mathcal{L}_{\text{inner},i}$ from θ_i, using coefficient c_i and minibatch $\mathcal{B}_{\text{in}}$. The resulting candidate parameters and policy are

$$\widetilde{\theta}_i(c_i) = \mathcal{U}_i(\theta_i; c_i, \mathcal{B}_{\text{in}}), \qquad \widetilde{\pi}_i = \pi_{\widetilde{\theta}_i(c_i)}. \tag{7}$$

The outer losses evaluate $\widetilde{\pi}_i$ on an independently drawn minibatch $\mathcal{B}_{\text{out}}$ and update c_i through Equation (6). Positive, detached scales $\hat{S}_i$ normalize the scores by mean absolute critic values at candidate actions. Appendix B.1 specifies the scales and stop-gradient operations; Appendix B.2 gives the optimizers and schedules.



(a) TD3+RAPO (b) IQL+RAPO

Figure 1: **Overview of RAPO.** π_β denotes the behavior represented by the offline data. (a) TD3+RAPO separates the bootstrap actor π_B from the execution actor π_E and adapts their coefficients α_B and α_E using target-stability and local-improvement objectives, respectively. (b) IQL+RAPO retains actor-independent Q/V learning from dataset transitions and adapts only the execution policy's inverse temperature β_E.

3.2 Execution Objective: Local Policy Improvement

The performance-difference identity relates return improvement to the candidate policy's action-value gain under Q^{π_E}, averaged over its discounted state distribution (Appendix A.2). RAPO approximates this criterion using a learned critic and dataset states, and scores the local change from the current action to the candidate action.

Write $\bar{Q}_E$ for the target critic used to score execution: $\bar{Q}_E = \bar{Q}_1$ in TD3+RAPO and $\bar{Q}_E = \bar{Q}_{\min}$ in IQL+RAPO. It is held fixed while scoring an update. Here π_E denotes the action map used in evaluation, including the mean for Gaussian actors.

For a state s, write $a_0 = \pi_E(s)$, $a_+ = \widetilde{\pi}_E(s)$, and $\delta a = a_+ - a_0$. A standard smoothness bound motivates an update term that rewards first-order improvement and a conservatism term that accounts for curvature.

Lemma 1 (Local critic improvement)**.** *Fix s. Suppose $\bar{Q}_E(s,\cdot)$ is differentiable on a neighborhood of the segment from a_0 to a_+ and its action gradient is L_s-Lipschitz on that segment. Then*

$$\bar{Q}_E(s,a_+) - \bar{Q}_E(s,a_0) \geq \nabla_a \bar{Q}_E(s,a_0)^\top \delta a - \frac{L_s}{2}\|\delta a\|_2^2. \tag{8}$$

To make the curvature penalty computable, define the detached endpoint gradients

$$g_0 = \mathrm{sg}(\nabla_a \bar{Q}_E(s,a_0)), \qquad g_+ = \mathrm{sg}(\nabla_a \bar{Q}_E(s,a_+)).$$

Under a local quadratic approximation with Hessian H_s, the endpoint-gradient difference satisfies $g_+ - g_0 = H_s \delta a$. Hence,

$$\left|\frac{1}{2}\delta a^\top H_s \delta a\right| \leq \frac{1}{2}\|g_+ - g_0\|_2 \|\delta a\|_2.$$

RAPO uses this computable penalty to discount the first-order update term, requiring two action gradients without forming a Hessian. This motivates the following surrogate score and execution loss:

$$\widehat{B}_\pi(s) = \underbrace{g_0^\top \delta a}_{\text{update}} - \underbrace{\frac{1}{2}\|g_+ - g_0\|_2 \|\delta a\|_2}_{\text{conservatism}}, \tag{9}$$

$$\mathcal{L}_E = -\frac{\mathbb{E}_{s\sim\mathcal{B}_{\text{out}}} \widehat{B}_\pi(s)}{\bar{S}_E}. \tag{10}$$

Minimizing $\mathcal{L}_E$ favors coefficients whose candidate updates have large first-order gains and small endpoint-gradient penalties. With g_0, g_+, and $\bar{S}_E$ detached, the coefficient gradient passes through δa. The penalty bounds the magnitude of a quadratic correction under the stated approximation; endpoint gradients alone do not establish a bound for a general critic. Appendix A.2 identifies the additional critic and state-distribution errors involved in relating this score to return improvement.

3.3 Bootstrap Objective: Target Stability

A bootstrap-policy update can move the critic's regression targets by changing the next-state actions in Equation (2). To isolate the effect of the policy update, we evaluate the current and candidate actions using the same fixed target critics.

For a transition $x = (s, a, r, s', d)$, let $d = 1$ indicate termination and define

$$y(\pi; x) = r + \gamma(1 - d)\bar{Q}_{\min}(s', \pi(s')).$$

The candidate-induced target change is

$$\begin{aligned}\Delta y_B(x) &= y(\widetilde{\pi}_B; x) - \mathrm{sg}(y(\pi_B; x)) \\ &= \gamma(1 - d)\left[\bar{Q}_{\min}(s', \widetilde{\pi}_B(s')) - \mathrm{sg}\big(\bar{Q}_{\min}(s', \pi_B(s'))\big)\right].\end{aligned} \tag{11}$$

We penalize its root-mean-square magnitude, so positive and negative target changes cannot cancel:

$$\mathcal{L}_B = \mathrm{sg}(\alpha_B)\sqrt{\mathbb{E}_{x\sim\mathcal{B}_{\mathrm{out}}}\left[\left(\frac{\Delta y_B(x)}{\bar{S}_B}\right)^2\right]}. \tag{12}$$

The critic loss in Equation (2) depends on values at bootstrap actions. With the target critics fixed, $\Delta y_B(x)$ measures how an unsmoothed Bellman target for the same transition responds to a candidate update of the online bootstrap actor. Penalizing its RMS favors updates with smaller target sensitivity while the inner TD3+BC loss still trains the actor for policy improvement. The actual training target uses a smoothed target actor (Appendix B.2), so this objective is a proxy for target stability. Algorithm 1 combines the outer objectives with the base updates.

Algorithm 1 RAPO (optimizer timing and schedules in Appendix B.2)

Input: Dataset $\mathcal{D}$; base method $m \in \{\mathrm{TD3+BC}, \mathrm{IQL}\}$; active roles $\mathcal{I}(m) = \{E, B\}$ or $\{E\}$, respectively.

Initialize actors, coefficients, and value/target networks.
for each training step **do**
  Sample $\mathcal{B}_{\mathrm{in}}$; apply the value updates of m from Section 2.2.
  if coefficient update is due **then**
    Sample independent $\mathcal{B}_{\mathrm{out}}$; form each role's candidate via (7).
    Update each c_i via (6), using $\mathcal{L}_E$ in (10) or $\mathcal{L}_B$ in (12).
  Apply the scheduled actor and target updates of m.
end for
Output: Trained value networks and actors $\{\pi_i : i \in \mathcal{I}(m)\}$; execute π_E.

4 Experimental Results

4.1 Evaluation Protocol

We evaluate TD3+RAPO and IQL+RAPO against TD3+BC, IQL, wPC (Peng et al., 2023), A2PR (Liu et al., 2024), and ASPC (Tan et al., 2026) on fifteen D4RL tasks (Fu et al., 2020): nine Locomotion tasks spanning HalfCheetah, Hopper, and Walker2d with medium, medium-replay, and medium-expert data, and six AntMaze tasks. The base actor-critic updates follow the CORL implementations (Tarasov et al., 2022), with network architectures and training settings detailed in Appendix B.

All final scores are measured at one million training steps: each of four independent training runs is evaluated over 50 episodes, and its episode mean gives one seed score. We report means and sample standard deviations across these four scores. Table 1 reports normalized returns for each task; domain and overall averages weight environments equally.

4.2 Benchmark Performance

TD3+RAPO averages 91.7 on Locomotion and 70.0 on AntMaze, compared with 80.8 and 31.7 for TD3+BC. IQL+RAPO averages 81.2 and 62.1, compared with 78.6 and 55.4 for IQL. Both RAPO

Table 1: D4RL performance at one million training steps, reported as mean ± sample standard deviation across four training seeds. Initial coefficients are fixed at five for both RAPO methods. Hyperparameters are reported in Appendix B.2. Average rows give equal weight to environments; the final row averages all fifteen tasks. Bold marks the highest mean in each row, including ties.

Dataset	TD3+BC	IQL	wPC	A2PR	ASPC	TD3+RAPO	IQL+RAPO
HalfCheetah-m	50.0±0.7	48.4±0.2	54.7±0.4	55.9±0.6	57.8±0.7	**58.2±0.5**	48.6±0.2
HalfCheetah-mr	46.2±0.5	44.2±0.4	48.2±0.4	48.0±1.1	**50.1±1.3**	49.7±0.7	44.6±0.5
HalfCheetah-me	99.5±1.0	89.5±3.9	100.0±0.5	98.9±1.7	92.2±12.8	**100.6±3.0**	92.8±0.6
Hopper-m	61.5±2.7	63.0±2.5	83.9±6.8	83.4±2.5	81.9±5.2	**101.8±0.2**	64.4±5.8
Hopper-mr	78.8±34.2	87.6±8.4	101.2±0.6	100.1±1.1	99.6±4.0	**101.5±0.6**	94.1±15.4
Hopper-me	111.4±2.0	99.8±24.0	103.5±7.4	**112.6±0.5**	109.2±2.7	111.0±1.3	108.3±5.1
Walker2d-m	83.7±2.8	81.4±3.5	88.9±1.4	89.3±1.3	89.5±15.4	**93.8±1.4**	83.2±2.2
Walker2d-mr	86.7±5.6	81.5±4.5	93.5±3.5	91.5±5.1	90.6±9.4	**98.0±1.4**	82.4±6.5
Walker2d-me	109.7±0.7	111.8±1.2	110.3±0.5	112.0±0.3	110.5±0.2	111.1±1.2	**112.5±0.8**
Locomotion average	80.8	78.6	87.1	88.0	86.8	**91.7**	81.2
Ant-umaze	90.0±8.2	76.0±14.0	**92.5±9.6**	**92.5±5.0**	88.0±3.7	**92.5±6.4**	77.5±10.8
Ant-umaze-diverse	87.5±5.0	58.0±5.9	80.0±8.2	55.0±41.2	82.5±4.7	**88.0±6.9**	63.0±7.0
Ant-medium-play	2.5±5.0	63.5±6.2	70.0±14.1	45.0±30.0	76.0±12.4	**81.5±3.0**	73.0±6.8
Ant-medium-diverse	10.0±8.2	66.0±13.1	60.0±14.1	47.5±9.6	49.5±22.3	59.0±17.2	**73.0±11.8**
Ant-large-play	0.0±0.0	35.0±10.4	32.5±5.0	15.0±12.9	**53.5±19.7**	48.0±6.9	45.5±5.7
Ant-large-diverse	0.0±0.0	34.0±11.0	45.0±17.3	15.0±17.3	32.5±20.9	**51.0±5.3**	40.5±4.4
AntMaze average	31.7	55.4	63.3	45.0	63.7	**70.0**	62.1
Overall average	61.2	69.3	77.6	70.8	77.6	**83.0**	73.5

variants exceed their respective baselines in the two domain averages, and TD3+RAPO has the highest overall average among the compared methods.

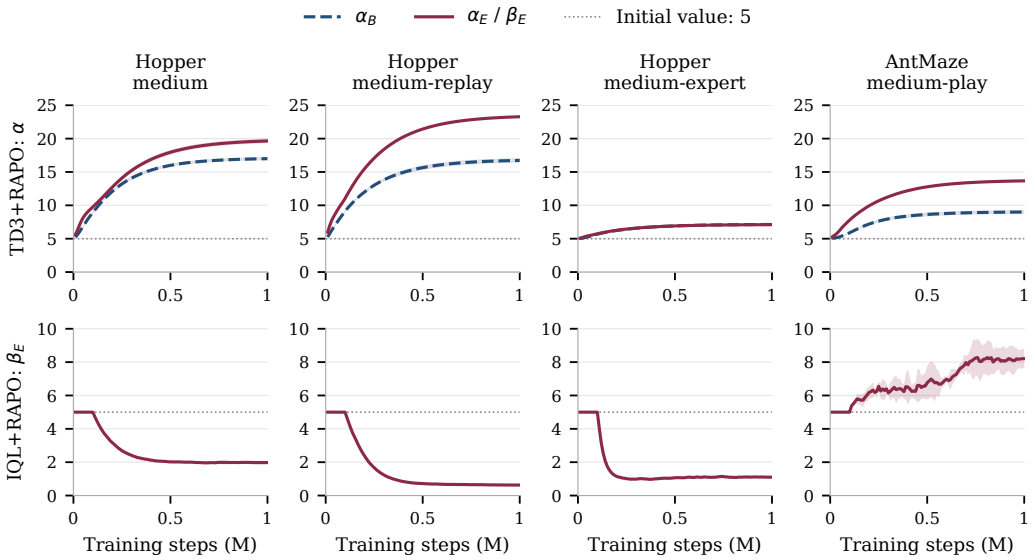


Figure 2: Learned coefficients on three Hopper datasets and AntMaze-medium-play. The top row shows TD3+RAPO's α_B and α_E; the bottom row shows IQL+RAPO's β_E. Curves and shaded bands show means and sample standard deviations across seeds. The dotted line marks the common initial value of five.

Learned coefficient trajectories. Figure 2 shows the coefficients learned from equal initial values. On Hopper-medium, Hopper-medium-replay, and AntMaze-medium-play, TD3+RAPO's α_E rises above α_B; on Hopper-medium-expert they remain close. In IQL+RAPO, β_E decreases on the three Hopper datasets and increases on AntMaze-medium-play. Larger α_i in TD3+RAPO weakens cloning relative to Q maximization, whereas larger β_E in IQL+RAPO increases the weight on higher-advantage dataset actions before clipping; their numerical values represent different scales.

4.3 Execution-Objective Ablation

With the actor structure and bootstrap objective held fixed, we compare $\widehat{B}_\pi$ with two simpler scores to test the criterion used for execution-coefficient adaptation.

Q uses the direct change in the fixed execution critic defined in Section 3.2, $f_Q(s) = \bar{Q}_E(s, a_+) - \mathrm{sg}(\bar{Q}_E(s, a_0))$, for the same candidate actions a_0, a_+.

Linear uses only the first-order gain from Equation (9), $f_{\mathrm{Linear}}(s) = g_0^\top \delta a$, omitting the endpoint-gradient penalty.

Each score defines an execution loss $-\mathbb{E}_{\mathcal{B}_{\mathrm{out}}}[f(s)]/\bar{S}_E$, using the normalization and detachment rules in Appendix B.1.

Q and Linear change only the execution score; they retain the two actors, RMS bootstrap objective, and coefficient learning rate of TD3+RAPO in each environment. All three conditions cover the fifteen benchmark tasks.

Table 2: Execution-score ablation at the selected coefficient learning rates. Q uses target-critic value change; Linear uses the first-order gain. Entries are equal-weight task means within each domain and overall. Bold marks the highest mean.

Domain	Q	Linear	$\widehat{B}_\pi$ (Ours)
Locomotion	87.0	88.7	**91.7**
AntMaze	66.6	65.5	**70.0**
Overall	78.8	79.4	**83.0**

Table 2 shows that the full score $\widehat{B}_\pi$ in $\mathcal{L}_E$ has the highest means in both domains and overall. Its overall mean is 83.0, compared with 78.8 for Q and 79.4 for Linear. The Linear control isolates the endpoint-gradient penalty; its lower averages suggest that this term helps select execution-coefficient updates. The full score also exceeds direct Q change under the same actor structure and bootstrap objective. These comparisons support $\mathcal{L}_E$ as an effective criterion for evaluating candidate execution updates in this setting.

4.4 Adaptation During Training

Bootstrap adaptation. To isolate bootstrap adaptation, we compare adapting α_B from five with fixing it at five. Both conditions retain two actors, adapt α_E from five, and use the same actor and critic settings and coefficient learning rate. Table 3 shows higher averages with bootstrap adaptation in both domains, particularly on AntMaze (55.8 to 70.0). The overall mean increases from 75.9 to 83.0. Gains are particularly large on AntMaze-large-play (20.0 to 48.0) and AntMaze-large-diverse (13.0 to 51.0). Appendix B.5 reports the per-task results and a fixed-$\alpha_B = 1$ control.

Table 3: Bootstrap-coefficient adaptation with α_E adapted in both conditions. The control fixes $\alpha_B = 5$; TD3+RAPO adapts it from five. Entries are equal-weight task means within each domain and overall. Per-task results appear in Table 8.

Domain	Fixed $\alpha_B = 5$	TD3+RAPO
Locomotion	89.3	91.7
AntMaze	55.8	70.0
Overall	75.9	83.0

Fixed learned endpoints. We next test whether fixed coefficients reproduce adaptation throughout training. For each environment, we retrain a two-actor agent with both coefficients fixed to the corresponding four-seed means of TD3+RAPO's final coefficients.

Table 4 shows a higher overall mean with adaptation (83.0 versus 78.4), with higher means in both domains. The per-task results in Appendix B.5 show that the Locomotion difference is concentrated in Hopper-medium-expert, while four of six AntMaze tasks improve. Learned final coefficient values therefore do not reproduce the adaptive domain averages when fixed from the start.

Table 4: Fixing learned endpoint coefficients from the start versus adapting from five. Both conditions retain separate TD3 actors. Scores are equal-weight task means; fixed constants and per-task results appear in Table 9.

Domain	Fixed endpoints	TD3+RAPO
Locomotion	88.7	91.7
AntMaze	62.9	70.0
Overall	78.4	83.0

5 Related Work

Behavior regularization. Offline RL must balance exploiting learned values against selecting actions outside dataset coverage (Levine et al., 2020). TD3+BC applies a quadratic BC penalty to the TD3 actor (Fujimoto et al., 2018; Fujimoto & Gu, 2021), while CQL regularizes value estimates toward pessimism (Kumar et al., 2020). ReBRAC combines architectural changes with separate regularization coefficients for actor learning and critic targets (Tarasov et al., 2023). These interventions act at different points in learning. RAPO retains the base value-loss form and adapts actor-loss coefficients, including the coefficient of the actor supplying bootstrap actions.

Adaptive constraints and meta-gradients. Weighted Policy Constraints favor desirable dataset actions (Peng et al., 2023), A2PR constructs an advantage-guided behavior reference (Liu et al., 2024), and selective regularization varies constraint strength across states (Luo et al., 2025). Meta-gradients learn update parameters by differentiating through optimization (Xu et al., 2018; Franceschi et al., 2018). ASPC uses this mechanism for policy-constraint scales, including IQL's inverse temperature (Tan et al., 2026). Its outer loss includes the squared change in mean Q, $(\mathbb{E}\Delta Q)^2$. RAPO instead assigns local improvement to execution and per-transition target RMS, $\sqrt{\mathbb{E}(\Delta y)^2}$, to bootstrapping. The latter prevents opposite-signed target changes from canceling. The distinction is the role-specific evaluation criterion, not the use of meta-gradients or value-change regularization itself.

Decoupling value learning and execution. Separating the policies involved in value learning and deployment has several precedents. One-step RL estimates the behavior policy's value and then performs a regularized improvement step without repeatedly evaluating the improved policy (Brandfonbrener et al., 2021). IQL and XQL learn values from dataset actions without requiring the extracted actor in their offline Bellman updates (Kostrikov et al., 2022; Garg et al., 2023). MCEP retains an explicit target policy and trains a separate, more mildly constrained evaluation policy (Xu et al., 2024). RAPO learns how strongly to regularize each actor by evaluating candidate updates according to the actor's role in execution or value learning.

Value learning and policy extraction. The extraction procedure determines how learned values are converted into executable actions. IDQL relates IQL's value objective to an implicit actor and uses critic-weighted samples from a diffusion behavior model for extraction (Hansen-Estruch et al., 2023). Experiments that vary value learning and extraction independently show that the extraction objective can substantially change performance even with the same value function (Park et al., 2024). Other decoupled methods rank behavior-model proposals at inference time (Lin et al., 2026). RAPO instead learns an explicit execution actor during training. Its IQL instantiation retains advantage-weighted regression and adapts its inverse temperature using a local critic-improvement score; it does not change the policy class or introduce inference-time search.

Local policy improvement. Performance-difference and trust-region analyses connect action-value improvement to return change under assumptions on the critic and state distribution (Schulman et al., 2015). RAPO uses a single candidate update to assess a coefficient, retaining the base actor objective. Its endpoint-gradient score is a local surrogate for this assessment; the conditional smoothness bound and the additional errors separating that score from return improvement are stated in Section 3.2 and Appendix A.

6 Conclusion

RAPO learns a policy-update coefficient by testing a candidate update against a criterion for the policy's use. In TD3+BC, this gives separate execution and bootstrap actors: the former is evaluated by a local critic-improvement surrogate, the latter by its effect on critic targets. In IQL, value updates already use dataset actions, so only the execution policy's extraction coefficient is adapted.

Both instantiations improve over their respective baselines in domain-average return on the fifteen D4RL tasks. Within the two-actor TD3 setting, the full execution score and adapting the bootstrap coefficient achieve higher domain averages than the tested controls; fixed learned endpoint coefficients also yield lower averages than continued adaptation. These findings motivate choosing coefficient-learning criteria according to how policy actions enter the algorithm. The execution score remains a local surrogate, and its relation to return improvement depends on the learned critic and the state distribution.

AI use statement

Generative AI tools assisted with discovering relevant literature; drafting, editing, and polishing portions of the manuscript; and developing and checking mathematical arguments, including the local critic-improvement lemma and its proof in Appendix A.1. The authors reviewed the AI-assisted work and take responsibility for the final text, citations, proofs, technical claims, and results.

References

David Brandfonbrener, Will Whitney, Rajesh Ranganath, and Joan Bruna. Offline RL Without Off-Policy Evaluation. In *Advances in Neural Information Processing Systems*, volume 34, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/274a10ffa06e434f2a94df765cac6bf4-Abstract.html.

Luca Franceschi, Paolo Frasconi, Saverio Salzo, Riccardo Grazzi, and Massimiliano Pontil. Bilevel Programming for Hyperparameter Optimization and Meta-Learning. In *ICML*, 2018. URL https://proceedings.mlr.press/v80/franceschi18a.html.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for Deep Data-Driven Reinforcement Learning. *arXiv preprint arXiv:2004.07219*, 2020. URL https://arxiv.org/abs/2004.07219.

Scott Fujimoto and Shixiang Shane Gu. A Minimalist Approach to Offline Reinforcement Learning. In *NeurIPS*, 2021. URL https://arxiv.org/abs/2106.06860.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *ICML*, 2018. URL https://proceedings.mlr.press/v80/fujimoto18a.html.

Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme Q-Learning: MaxEnt RL without Entropy. In *ICLR*, 2023. URL https://arxiv.org/abs/2301.02328.

Philippe Hansen-Estruch, Ilya Kostrikov, Michael Janner, Jakub Grudzien Kuba, and Sergey Levine. IDQL: Implicit Q-Learning as an Actor-Critic Method with Diffusion Policies. *arXiv preprint arXiv:2304.10573*, 2023. URL https://arxiv.org/abs/2304.10573.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline Reinforcement Learning with Implicit Q-Learning. In *ICLR*, 2022. URL https://arxiv.org/abs/2110.06169.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning. In *NeurIPS*, 2020. URL https://arxiv.org/abs/2006.04779.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv preprint arXiv:2005.01643*, 2020. URL https://arxiv.org/abs/2005.01643.

Xuyao Lin, Yixiang Shan, Jinru Duan, Tao Yang, Xinyu Zhao, Runyu Lei, Yiming Zhao, Jiaxin Fan, Zongbao Feng, and Peng Jia. Decoupling Policy Extraction for Offline Reinforcement Learning. *arXiv preprint arXiv:2608.20909*, 2026. URL https://arxiv.org/abs/2608.20909.

Tenglong Liu, Yang Li, Yixing Lan, Hao Gao, Wei Pan, and Xin Xu. Adaptive Advantage-Guided Policy Regularization for Offline Reinforcement Learning. In *ICML*, 2024. URL https://proceedings.mlr.press/v235/liu24ai.html.

Qin-Wen Luo, Ming-Kun Xie, Ye-Wen Wang, and Sheng-Jun Huang. Learning to Trust Bellman Updates: Selective State-Adaptive Regularization for Offline RL. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41450–41467, 2025. URL https://proceedings.mlr.press/v267/luo25p.html.

Seohong Park, Kevin Frans, Sergey Levine, and Aviral Kumar. Is Value Learning Really the Main Bottleneck in Offline RL? In *NeurIPS*, 2024. URL https://arxiv.org/abs/2406.09329.

Zhiyong Peng, Changlin Han, Yadong Liu, and Zongtan Zhou. Weighted Policy Constraints for Offline Reinforcement Learning. In *AAAI*, 2023. URL https://ojs.aaai.org/index.php/AAAI/article/view/26130.

John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. Trust Region Policy Optimization. In *ICML*, 2015. URL https://proceedings.mlr.press/v37/schulman15.html.

Jing Tan, Xiaorui Li, Chao Yao, Xiaojuan Ban, Yuetong Fang, Renjing Xu, and Zhaolin Yuan. Adaptive Scaling of Policy Constraints for Offline Reinforcement Learning. In *ICLR*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/713939ee5a7b14a6fecf69e692ec3799-Abstract-Conference.html.

Denis Tarasov, Alexander Nikulin, Dmitry Akimov, Vladislav Kurenkov, and Sergey Kolesnikov. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. URL https://openreview.net/forum?id=SyAS49bBcv.

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the Minimalist Approach to Offline Reinforcement Learning. In *NeurIPS*, 2023. URL https://arxiv.org/abs/2305.09836.

Linjie Xu, Zhengyao Jiang, Jinyu Wang, Lei Song, and Jiang Bian. Mildly Constrained Evaluation Policy for Offline Reinforcement Learning. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=imAROs79Pb.

Zhongwen Xu, Hado van Hasselt, and David Silver. Meta-Gradient Reinforcement Learning. In *NeurIPS*, 2018. URL https://arxiv.org/abs/1805.09801.

A Analysis of the Execution Objective

A.1 Proof of Lemma 1

Proof. Fix s and write $f(a) = \bar{Q}_E(s, a)$. The fundamental theorem of calculus and the Lipschitz condition give

$$\begin{aligned} f(a_+) - f(a_0) - \nabla f(a_0)^\top \delta a &= \int_0^1 \left[\nabla f(a_0 + t\delta a) - \nabla f(a_0)\right]^\top \delta a \, \mathrm{d}t \\ &\geq -\int_0^1 L_s t \|\delta a\|_2^2 \, \mathrm{d}t = -\frac{L_s}{2}\|\delta a\|_2^2. \end{aligned}$$

□

The smoothness premise can fail at activation boundaries or, for IQL's minimum critic, where the minimizing Q-network switches.

Quadratic interpretation of the conservatism term. If $\bar{Q}_E(s, a)$ is quadratic in a with symmetric Hessian H_s, then $g_+ - g_0 = H_s \delta a$ and

$$\begin{aligned} \bar{Q}_E(s, a_+) - \bar{Q}_E(s, a_0) &= g_0^\top \delta a + \tfrac{1}{2}\delta a^\top H_s \delta a, \\ \left|\tfrac{1}{2}\delta a^\top H_s \delta a\right| &\leq \tfrac{1}{2}\|H_s \delta a\|_2 \|\delta a\|_2 = \tfrac{1}{2}\|g_+ - g_0\|_2 \|\delta a\|_2. \end{aligned}$$

Thus Equation (9) subtracts an upper bound on the magnitude of the quadratic correction in this special case. For a general critic, endpoint variation need not control interior curvature; the score is not a certified lower bound. This curvature penalty also does not quantify uncertainty in the critic.

A.2 Relation to Return Improvement

For deterministic policies π and π^+ with a common initial-state distribution, the performance-difference identity is (Schulman et al., 2015)

$$J(\pi^+) - J(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{s\sim d^{\pi^+}}\left[Q^\pi(s, \pi^+(s)) - Q^\pi(s, \pi(s))\right], \tag{13}$$

where d^{π^+} is normalized discounted state occupancy. To relate this identity to RAPO, set $\pi = \pi_E$, $\pi^+ = \widetilde{\pi}_E$, let ν be the dataset state distribution, and define $\Delta_E(s) = \bar{Q}_E(s, \pi^+(s)) - \bar{Q}_E(s, \pi(s))$. Assuming integrability, adding and subtracting Δ_E gives the exact decomposition

$$\begin{aligned} (1-\gamma)\left[J(\pi^+) - J(\pi)\right] &= \mathbb{E}_\nu \widehat{B}_\pi + \mathcal{R}_{\text{local}} + \mathcal{R}_{\text{critic}} + \mathcal{R}_{\text{state}}, \\ \mathcal{R}_{\text{local}} &= \mathbb{E}_\nu\left[\Delta_E - \widehat{B}_\pi\right], \\ \mathcal{R}_{\text{critic}} &= \mathbb{E}_{d^{\pi^+}}\left[Q^\pi(s, \pi^+(s)) - Q^\pi(s, \pi(s)) - \Delta_E(s)\right], \\ \mathcal{R}_{\text{state}} &= \mathbb{E}_{d^{\pi^+}}\Delta_E - \mathbb{E}_\nu \Delta_E. \end{aligned} \tag{14}$$

These residuals capture the endpoint approximation, critic error relative to Q^π (including reference-policy mismatch), and the use of dataset states. RAPO does not bound them. Even if the learned critic is smooth, endpoint gradients need not control interior curvature; hence the execution score does not guarantee return improvement.

B Algorithm Instantiations and Implementation Details

B.1 Differentiation and Normalization

The optimizer updates an unconstrained scalar ρ_i for each active role, with the coefficient c_i from Section 3.1:

$$\begin{aligned} &\text{TD3+RAPO:} \quad c_i = \alpha_i = \text{softplus}(\rho_i), \quad i \in \{B, E\}, \qquad \frac{\mathrm{d}\mathcal{L}_i}{\mathrm{d}\rho_i} = \frac{\mathrm{d}c_i}{\mathrm{d}\rho_i}\frac{\mathrm{d}\mathcal{L}_i}{\mathrm{d}c_i}. \\ &\text{IQL+RAPO:} \quad c_E = \beta_E = \exp(\rho_E), \end{aligned}$$

Each ρ_i has a separate Adam state. The second factor follows Equation (6) with the specified stop-gradient operations.

The candidate step leaves real actor parameters and optimizer state unchanged. Starting actor parameters, critic parameters, dataset-action references, and incoming optimizer states are constants for coefficient differentiation; new gradients and optimizer moments remain differentiable in c_i. Detached quantities are recomputed at each evaluation.

Using the critics from Sections 3.2 and 3.3, the detached value scales are defined as follows, with $\epsilon = 10^{-6}$:

$$\bar{S}_E = \mathrm{sg}\big(\max\left\{\mathbb{E}_{\mathcal{B}_{\mathrm{out}}}|\bar{Q}_E(s, \widetilde{\pi}_E(s))|, \epsilon\right\}\big)\,, \tag{15}$$

$$\bar{S}_B = \mathrm{sg}\big(\mathbb{E}_{\mathcal{B}_{\mathrm{out}}}|\bar{Q}_{\min}(s, \widetilde{\pi}_B(s))|\big) + \epsilon. \tag{16}$$

The bootstrap prefactor $\mathrm{sg}(\alpha_B)$ scales the coefficient gradient without adding a direct derivative path; dependence on α_B enters through the candidate policy. The bootstrap RMS includes terminal transitions as zeros and has no additive epsilon inside the square root; its value and chosen subgradient are zero when all inputs are zero.

B.2 Training Settings

The actor-critic implementations follow CORL (Tarasov et al., 2022). For the TD3+BC, wPC, and ASPC comparisons, we use the same Q-network architecture as TD3+RAPO: three hidden layers of width 256 with ReLU activations and LayerNorm in each hidden layer.

Both actual and candidate TD3 actor updates use the loss scale in Equation (1); multiplying it by α_i changes finite updates and their meta-gradients. The training critic uses the target bootstrap actor with clipped Gaussian action noise ($\sigma = 0.2$, $c = 0.5$), whereas Equation (11) compares unsmoothed online actors. Both methods mask the next-state value by $1 - d$, where d indicates termination.

IQL clips $\exp(\beta_E A)$ at 100. Gaussian actor means are used in the execution score and at evaluation.

Table 5 summarizes RAPO's training settings. At initialization five, we select each method's coefficient learning rate from $\{3 \times 10^{-4}, 10^{-3}, 2 \times 10^{-3}\}$ by its four-seed mean in each environment. Table 6 lists the selected rates.

Table 5: Training settings. Intervals are measured in value-update steps; $\bar{Q} \leftarrow (1 - \tau_Q)\bar{Q} + \tau_Q Q$ defines the target-update rate.

Setting	TD3+RAPO	IQL+RAPO
Actor hidden layers	2×256	2×256
Value hidden layers	Q: 3×256, LayerNorm	Q and V: 2×256, no LayerNorm
Network optimizer / initial rate	Adam / 3×10^{-4}	Adam / 3×10^{-4}
Actor learning-rate schedule	Constant	Cosine decay
Candidate optimizer	E: current-state Adam; B: SGD	E: current-state Adam
Actual actor optimizer / interval	Adam / 2	Adam / 1
Coefficient initialization	$\alpha_B = \alpha_E = 5$	$\beta_E = 5$
Coefficient update interval	20	20, after 100,000 steps
Coefficient rate schedule	Exponential to 0.01 of initial rate	Constant
Coefficient range	$\alpha_i > 0$	$\beta_E \in [0.05, 100]$
Target-update rate τ_Q	0.005	0.005

Candidate execution steps use the actor's current Adam moments and learning rate. The bootstrap candidate uses SGD at the actor learning rate; the actual bootstrap update uses Adam. Network Adam uses moments $(0.9, 0.999)$ and epsilon 10^{-8}. TD3 coefficient Adam uses the same moments; IQL coefficient Adam uses $(0, 0.999)$, with ρ_E projected to $[\log 0.05, \log 100]$ after each update without resetting its moments.

Table 6: Main-selected coefficient learning rates with initialization five. TD3 uses the listed rate for both coefficients; its fixed-bootstrap control reuses the same rate for α_E. Fixed-endpoint TD3 has no active coefficient update.

Environment	TD3+RAPO	IQL+RAPO
HalfCheetah-medium	2×10^{-3}	3×10^{-4}
HalfCheetah-medium-replay	2×10^{-3}	10^{-3}
HalfCheetah-medium-expert	10^{-3}	10^{-3}
Hopper-medium	2×10^{-3}	3×10^{-4}
Hopper-medium-replay	2×10^{-3}	3×10^{-4}
Hopper-medium-expert	3×10^{-4}	10^{-3}
Walker2d-medium	10^{-3}	3×10^{-4}
Walker2d-medium-replay	2×10^{-3}	10^{-3}
Walker2d-medium-expert	2×10^{-3}	10^{-3}
AntMaze-umaze	3×10^{-4}	10^{-3}
AntMaze-umaze-diverse	3×10^{-4}	3×10^{-4}
AntMaze-medium-play	10^{-3}	2×10^{-3}
AntMaze-medium-diverse	3×10^{-4}	3×10^{-4}
AntMaze-large-play	10^{-3}	2×10^{-3}
AntMaze-large-diverse	10^{-3}	10^{-3}

For both instantiations, the actual execution update uses the coefficient cached before that step's meta-update. TD3 first updates its critics, then, on scheduled steps, updates ρ_E and ρ_B, applies both real actor updates (using the new α_B), and updates its target networks. The TD3 bootstrap target actor uses $\bar{\theta}_B \leftarrow (1 - \tau_Q)\bar{\theta}_B + \tau_Q \theta_B$ with the same rate as its target critics. IQL caches the advantage and next-state value, updates V, Q, and target Q, performs the scheduled execution meta-update, and applies its real execution update.

Both instantiations use batch size 256, discount 0.99, normalized observations, and unchanged Locomotion rewards. TD3's smoothed target actions are clipped to $[-1, 1]$.

IQL+RAPO uses Gaussian actors and an expectile of 0.7 on Locomotion and 0.9 on AntMaze. AntMaze rewards are shifted by -1. Actor means use tanh, and log standard deviations are state-independent and bounded in $[-20, 2]$.

IQL baseline. The benchmark baseline uses the same actor family, target-update rate, expectiles, and reward handling, with fixed $\beta = 3$ on Locomotion and $\beta = 10$ on AntMaze.

B.3 Evaluation and Configuration Selection

The final-score and uncertainty protocol is defined in Section 4.1. For domain uncertainty, environments are averaged within each seed before computing the sample standard deviation across seeds.

B.4 Coefficient Learning-Rate Sensitivity

Table 7 tests TD3+RAPO with each initial coefficient learning rate shared across environments; other settings follow Table 5. It exceeds TD3+BC's domain averages at all three rates, though task-level performance varies. At 3×10^{-4}, its Locomotion and AntMaze means are 87.4 and 64.2, compared with 80.8 and 31.7 for TD3+BC.

B.5 Outer-Objective and Coefficient Ablations

Bootstrap adaptation at main-selected settings. Table 8 extends the fixed-$\alpha_B = 5$ comparison to fixed and adaptive values of one. All four conditions retain two actors, adapt α_E from five, and use TD3+RAPO's actor and critic settings and selected coefficient learning rate.

Table 7: TD3+RAPO sensitivity to the initial coefficient learning rate, with coefficient initialization five. Average rows weight environments equally.

Environment	3×10^{-4}	10^{-3}	2×10^{-3}
HalfCheetah-medium	54.3 ± 0.6	56.4 ± 0.7	58.2 ± 0.5
HalfCheetah-medium-replay	47.9 ± 0.5	49.3 ± 0.3	49.7 ± 0.7
HalfCheetah-medium-expert	98.1 ± 2.8	100.6 ± 3.0	98.0 ± 3.9
Hopper-medium	83.9 ± 11.2	96.4 ± 5.7	101.8 ± 0.2
Hopper-medium-replay	96.7 ± 6.4	99.8 ± 2.6	101.5 ± 0.6
Hopper-medium-expert	111.0 ± 1.3	79.1 ± 20.3	45.4 ± 5.5
Walker2d-medium	89.6 ± 1.7	93.8 ± 1.4	88.7 ± 12.7
Walker2d-medium-replay	95.0 ± 2.5	94.0 ± 4.5	98.0 ± 1.4
Walker2d-medium-expert	110.0 ± 0.5	110.8 ± 0.7	111.0 ± 1.2
Locomotion Avg.	87.4	86.7	83.6
AntMaze-umaze	92.5 ± 6.4	88.0 ± 4.3	85.0 ± 7.4
AntMaze-umaze-diverse	88.0 ± 6.9	35.5 ± 29.9	77.0 ± 12.7
AntMaze-medium-play	72.5 ± 14.9	81.5 ± 3.0	71.0 ± 9.5
AntMaze-medium-diverse	59.0 ± 17.2	43.0 ± 20.0	26.5 ± 19.8
AntMaze-large-play	29.5 ± 13.4	48.0 ± 6.9	46.5 ± 12.6
AntMaze-large-diverse	43.5 ± 5.3	51.0 ± 5.3	30.0 ± 14.6
AntMaze Avg.	64.2	57.8	56.0

Figure 3 extends the domain summary in Table 3. Adapting α_B from one raises the Locomotion mean from 88.2 to 90.9 and the AntMaze mean from 23.8 to 38.5 relative to fixing it at one. Bootstrap adaptation therefore improves both domain averages at either initial value. Initialization still matters, especially on AntMaze, where adaptation from five reaches 70.0 compared with 38.5 from one.

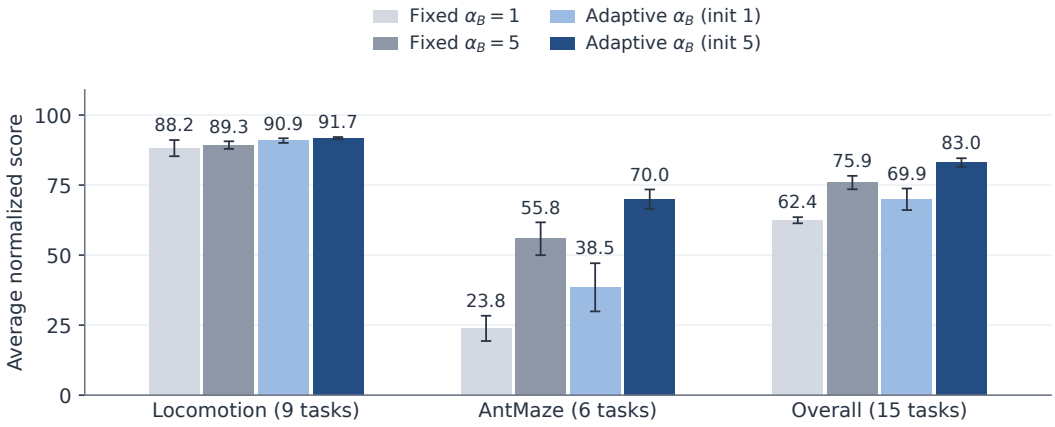


Figure 3: Fixed and adaptive bootstrap regularization at TD3+RAPO's selected learning rates. Bars show domain or overall task averages; error bars show standard deviations across seed-wise task averages. All conditions adapt α_E from five; adaptive α_B starts at one or five. Per-task results appear in Table 8.

Table 8: Fixed and adaptive bootstrap regularization at the selected learning rates. All conditions retain two actors and adapt α_E from five. Under Fixed, column numbers are constant α_B values; under Adaptive, they are initial values.

Environment	Fixed α_B		Adaptive α_B	
	1	5	1	5
HalfCheetah-medium	56.6 ± 0.9	59.1 ± 0.6	58.1 ± 1.0	58.2 ± 0.5
HalfCheetah-medium-replay	48.7 ± 2.4	50.0 ± 0.6	49.1 ± 0.5	49.7 ± 0.7
HalfCheetah-medium-expert	103.1 ± 3.6	98.8 ± 3.7	95.3 ± 5.7	100.6 ± 3.0
Hopper-medium	88.5 ± 24.2	101.7 ± 0.5	101.9 ± 0.7	101.8 ± 0.2
Hopper-medium-replay	98.7 ± 1.3	101.3 ± 1.2	100.5 ± 1.5	101.5 ± 0.6
Hopper-medium-expert	111.0 ± 2.3	99.1 ± 10.1	112.5 ± 0.2	111.0 ± 1.3
Walker2d-medium	85.0 ± 0.2	90.3 ± 3.1	93.2 ± 1.8	93.8 ± 1.4
Walker2d-medium-replay	91.9 ± 1.8	92.6 ± 9.7	96.5 ± 3.6	98.0 ± 1.4
Walker2d-medium-expert	110.2 ± 0.8	110.6 ± 0.8	110.9 ± 0.8	111.1 ± 1.2
Locomotion average	88.2	89.3	90.9	91.7
AntMaze-umaze	84.5 ± 4.4	90.0 ± 4.3	90.0 ± 1.6	92.5 ± 6.4
AntMaze-umaze-diverse	58.5 ± 26.3	88.5 ± 12.0	57.5 ± 39.5	88.0 ± 6.9
AntMaze-medium-play	0.0 ± 0.0	62.0 ± 5.4	46.5 ± 17.0	81.5 ± 3.0
AntMaze-medium-diverse	0.0 ± 0.0	61.5 ± 16.3	6.5 ± 6.0	59.0 ± 17.2
AntMaze-large-play	0.0 ± 0.0	20.0 ± 4.3	24.5 ± 15.3	48.0 ± 6.9
AntMaze-large-diverse	0.0 ± 0.0	13.0 ± 8.9	6.0 ± 9.5	51.0 ± 5.3
AntMaze average	23.8	55.8	38.5	70.0

Fixed learned endpoints. For each environment, we use the mean final coefficients of the four TD3+RAPO seeds as a fixed pair, rounded to six decimal places for training. We train new actors and critics with coefficient updates disabled. Table 9 displays the constants and per-task results; the adaptive column uses the main benchmark.

Table 9: Endpoint-fixed TD3 control. Each coefficient pair averages the final values of the main runs and remains fixed throughout the control runs. Constants are displayed to one decimal place; training uses six decimal places. TD3+RAPO uses the main benchmark runs.

Environment	Fixed α_E	Fixed α_B	Fixed endpoints	TD3+RAPO
HalfCheetah-medium	19.4	13.4	58.3 ± 0.4	58.2 ± 0.5
HalfCheetah-medium-replay	18.8	17.4	49.9 ± 0.9	49.7 ± 0.7
HalfCheetah-medium-expert	12.6	10.2	98.8 ± 5.4	100.6 ± 3.0
Hopper-medium	19.6	17.0	101.9 ± 0.4	101.8 ± 0.2
Hopper-medium-replay	23.3	16.7	101.0 ± 1.1	101.5 ± 0.6
Hopper-medium-expert	7.1	7.1	83.8 ± 20.3	111.0 ± 1.3
Walker2d-medium	11.8	9.8	95.5 ± 6.8	93.8 ± 1.4
Walker2d-medium-replay	20.7	18.2	98.2 ± 2.6	98.0 ± 1.4
Walker2d-medium-expert	19.3	12.8	111.1 ± 0.5	111.1 ± 1.2
AntMaze-umaze	7.8	6.7	95.5 ± 3.4	92.5 ± 6.4
AntMaze-umaze-diverse	7.5	6.3	61.5 ± 35.9	88.0 ± 6.9
AntMaze-medium-play	13.7	9.0	75.5 ± 9.1	81.5 ± 3.0
AntMaze-medium-diverse	7.7	6.0	49.0 ± 17.3	59.0 ± 17.2
AntMaze-large-play	13.5	9.2	53.0 ± 6.8	48.0 ± 6.9
AntMaze-large-diverse	13.7	9.3	43.0 ± 13.2	51.0 ± 5.3