Trust Region Q Adjoint Matching

Yonghoon Dong[1] **Kyungmin Lee**[1] **Changyeon Kim**[1]
Jaehyuk Kim[1] **Jinwoo Shin**[1,2]
[1]KAIST AI [2]RLWRLD
yonghoon.dong@kaist.ac.kr

Abstract

Off-policy reinforcement learning of pretrained flow policies remains challenging due to the instability of optimization arising from the multi-step sampling process. Recently, Q-learning with Adjoint Matching (QAM) addressed this by recasting policy improvement as a stochastic optimal control (SOC) problem guided by a learned critic. However, QAM inherits a fundamental fragility of critic-guided improvement, since small critic errors can be exponentially amplified and often lead to performance collapse. This paper introduces Trust Region Q Adjoint Matching (TRQAM), a stable off-policy fine-tuning algorithm that adaptively controls the path-space KL between the fine-tuned and pretrained policies through projected dual descent. Specifically, we adapt a trust-region parameter λ inside the sampling process of the flow policy and prove that λ exactly weights the path-space KL in the SOC objective. As a result, our method can tightly control the deviation from the pretrained policy within a prescribed KL budget, achieving stable off-policy RL. Through experiments on 50 OGBench tasks, TRQAM consistently outperforms prior methods in both offline RL and offline-to-online RL. In particular, TRQAM achieves an overall success rate of 68% in offline RL, substantially improving over the strongest baseline at 46%.

Code: github.com/yonghdong/trqam Blog: yonghdong.github.io/blog/trqam

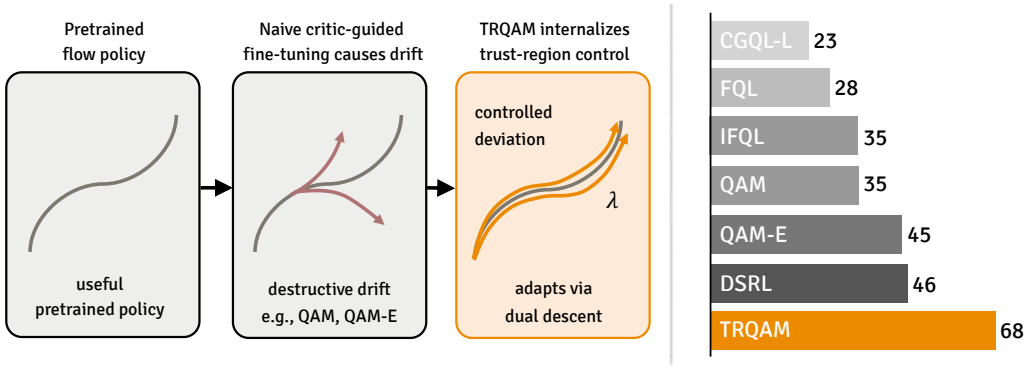
1 Introduction

Recently, flow matching policies [2, 6, 26, 29] have emerged as a promising approach to model rich and diverse action distributions, enabling high-capacity behavior generation beyond conventional uni-modal Gaussian policies. A pretrained flow policy captures useful skills, behavioral constraints, and broad coverage over plausible actions, making it an attractive prior for downstream off-policy RL fine-tuning [37, 10, 17, 44, 45, 18, 20, 49, 52, 56, 62, 65].

However, as the flow policy is defined implicitly through a multi-step denoising process, gradient-based policy improvement requires differentiating through the multi-step sampling chain, making direct backpropagation expensive and unstable [41, 64]. Existing approaches sidestep this through residual-style methods that keep the pretrained policy frozen and learn an additive residual to its actions [10, 60], or noise-space RL methods that freeze the pretrained flow policy and run actor-critic over its input noise [56]. The improvement is thus carried by an auxiliary module instead of the generative model itself, so these methods either compromise the expressiveness of the flow policy or cannot go beyond what the frozen pretrained policy already generates.

Recently, Q-learning with Adjoint Matching (QAM) [24] addressed this by recasting fine-tuning as a memoryless stochastic optimal control (SOC) problem. QAM uses a learned critic to steer the sampling process toward higher-value actions via Adjoint Matching. While this resolves the multi-step sampling instability, critic-induced instability still remains. In off-policy RL, the learned critic is inevitably imperfect. Its approximation errors compound through TD bootstrapping and produce systematic overestimation [12]. Critic-guided policy updates can then amplify these errors

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Higher reward, without destroying the pretrained policy (b) Offline RL success rate on OGBench

Figure 1: **TRQAM builds adaptive trust-region control into the SOC sampling process. (a):** Each curve is a sampling trajectory of the flow policy, from noise X_0 to action X_1. Fixed-temperature methods such as QAM and QAM-E [24] suffer from destructive drift, where small critic errors can be exponentially amplified into large policy errors (Lemma 1). TRQAM instead bounds the deviation from the pretrained policy through a trust-region parameter λ internalized in the SOC sampling process, which limits how far critic errors can move the policy. **(b):** Offline RL success rate on 50 OGBench [40] tasks, where TRQAM outperforms all baselines.

into large policy errors (see Lemma 1). QAM [24] acknowledges this and applies gradient clipping as a partial remedy, while calling for a more principled method beyond this heuristic. We empirically observe that gradient clipping does not prevent this fragility, as the adjoint loss can become unstable and collapse task performance on Robomimic [32] (Figures 2, 11 and 12). Thus, the central challenge becomes how to improve downstream performance without destructive drift from the pretrained prior, similar to the trust-region principle in on-policy RL [50, 51].

We propose *Trust Region Q Adjoint Matching* (TRQAM), an algorithm for stable off-policy fine-tuning of pretrained flow policies. TRQAM introduces a trust-region parameter λ directly into the SOC sampling process and adapts it via projected dual descent to enforce a prescribed KL bound between the fine-tuned and pretrained policies. Our central theoretical result, proved using Girsanov's theorem, shows that letting λ scale the control term in the drift makes λ the exact weight of the path-space KL in the SOC objective. Unlike a conventional KL regularizer added to the loss, this weight lives inside the SOC sampling process, so the dual update enforces the target KL bound through the SOC formulation itself.

This distinction matters in practice. Conventional loss-level KL regularization only competes with critic guidance at the loss level, allowing strong critic signals to push the realized KL far beyond the target bound and leave the policy vulnerable to collapse. In contrast, TRQAM tightly tracks the target bound throughout both offline and online training (Figures 6 and 15 to 17). The prescribed target KL bound therefore provides a practical control over how much the fine-tuned policy can deviate from the pretrained policy, and its best setting varies systematically with task structure (Section 4.3). Through experiments on 50 OGBench tasks, TRQAM consistently outperforms prior methods in both offline RL and offline-to-online RL. In particular, TRQAM achieves an overall offline RL success rate of 68%, substantially outperforming the strongest baseline at 46% (Table 1).

Contributions. We highlight the key contributions of our paper below:

- We identify the exponential amplification of critic errors as a fundamental fragility of fixed-temperature Adjoint Matching, formalized by Lemma 1 and confirmed empirically on `Robomimic`.
- We prove via Girsanov (Theorem 1) that letting λ scale the control term makes λ the exact weight of the path-space KL in the SOC objective, turning λ into a principled trust-region parameter.
- We propose *Trust Region Q Adjoint Matching* (TRQAM), which internalizes λ inside the SOC sampling process and adapts it via projected dual descent to enforce a target KL bound at the sampling level rather than as a loss-level penalty.
- On 50 OGBench tasks, TRQAM consistently outperforms prior methods on offline RL and offline-to-online RL. In particular, TRQAM achieves an overall success rate of 68% in offline RL, compared to 46% for the strongest baseline.

2 Background

Reinforcement learning. We consider a Markov decision process (MDP) [53] $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$ with state space $\mathcal{S}$, continuous action space $\mathcal{A} \subseteq \mathbb{R}^d$, transition kernel $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, discount factor $\gamma \in [0, 1)$, and initial state distribution $\rho_0 \in \Delta(\mathcal{S})$, where $\Delta(\cdot)$ denotes the set of probability distributions. The goal is to learn a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ that maximizes the expected discounted return $\mathbb{E}_{\rho_0,\pi,P}\big[\sum_{t\geq 0} \gamma^t r(s_t, a_t)\big]$. To this end, we learn a critic $Q^\pi(s, a) = \mathbb{E}_\pi\big[\sum_{k\geq 0} \gamma^k r(s_{t+k}, a_{t+k}) \mid s_t = s, a_t = a\big]$ that guides policy improvement.

We focus on off-policy fine-tuning of a pretrained flow policy $\pi_{\text{base}}(\cdot \mid s)$. Training draws transitions (s, a, r, s') from a replay buffer $\mathcal{D}$. The buffer initially holds a dataset collected by a possibly unknown or suboptimal behavior policy, and π_{base} is pretrained on it by behavior cloning. Fine-tuning first runs offline on this dataset alone. It then continues online, where $\mathcal{D}$ grows with the rollouts of the fine-tuned policy π_θ.

Flow matching and flow policy. Flow Matching [2, 26] learns a velocity field $v_\theta(x, \tau)$ that transports samples from a simple source distribution $p_0 = \mathcal{N}(0, I_d)$ to a target distribution p_1 through an ODE:

$$dX_\tau = v_\theta(X_\tau, \tau)\, d\tau, \qquad X_0 \sim p_0, \quad X_1 \sim p_1.$$

A standard choice is the optimal transport (OT) path, with straight-line interpolation $X_\tau = (1 - \tau)X_0 + \tau X_1$ between endpoint pairs $(X_0, X_1) \sim p_0 \otimes p_1$ and target velocity $\mathbb{E}[X_1 - X_0 | X_\tau = x]$. Then, the velocity field is trained by regressing v_θ against the target velocity $X_1 - X_0$, giving

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{\tau, X_0, X_1}\Big[\big\|v_\theta(X_\tau, \tau) - (X_1 - X_0)\big\|^2\Big].$$

A flow policy $\pi_\theta(\cdot \mid s)$ applies flow matching to action generation, sampling an action X_1 for a state s by integrating $dX_\tau = v_\theta(X_\tau, \tau; s)\, d\tau$ from $X_0 \sim \mathcal{N}(0, I_d)$.

Stochastic optimal control and Adjoint Matching. Stochastic optimal control (SOC) is a framework that fine-tunes a pretrained flow model by steering its sampling process with a learned control u toward low terminal cost. Training the model directly would require backpropagating through the multi-step sampling chain, which is costly and unstable, much like backpropagation through time in RNNs [41, 64]. Domingo-Enrich et al. [9] showed that the SOC objective can instead be solved with a lean adjoint ODE, which avoids backpropagating through the sampling chain and yields the Adjoint Matching algorithm.

Concretely, the flow ODE is replaced with an SDE that has the same marginal distribution at every $\tau \in [0, 1]$ [31], so its terminal X_1 still follows π_{base}. Since this SDE is not unique, Adjoint Matching fixes its diffusion coefficient to $\sqrt{2(1 - \tau)/\tau}$, which it calls the memoryless OT schedule. Under this schedule, X_0 is independent of X_1, so the optimum of the SOC objective below exponentially tilts π_{base} toward low terminal cost. Following Domingo-Enrich et al. [8], we write this coefficient as $\sqrt{\lambda}\,\sigma(\tau)$. Because the coefficient itself is fixed, λ enters only through $\sigma(\tau) = \sqrt{2(1 - \tau)/(\lambda\tau)}$, which scales the control term $\sigma(\tau)\, u$ in the drift. This gives the base and controlled SDEs

$$dX^{\text{base}}_\tau = b(X^{\text{base}}_\tau, \tau)\, d\tau + \sqrt{\lambda}\,\sigma(\tau)\, dB_\tau, \tag{1}$$

$$dX^u_\tau = \big(\, b(X^u_\tau, \tau) + \sigma(\tau)\, u(X^u_\tau, \tau)\,\big)\, d\tau + \sqrt{\lambda}\,\sigma(\tau)\, dB_\tau, \tag{2}$$

where $b(x, \tau) = 2\, v^{\text{base}}(x, \tau) - x/\tau$ is the drift given by the pretrained velocity field v^{base} under the memoryless OT schedule [9]. The base SDE generates the pretrained policy π_{base}. The control u modifies this process, and its magnitude determines how far the resulting policy moves from π_{base}.

Given a terminal cost $g : \mathbb{R}^d \to \mathbb{R}$, SOC seeks the control u by optimizing

$$\min_u \; \mathbb{E}_{\mathbf{X}\sim\mathbb{P}^u}\left[\frac{1}{2}\int_0^1 \big\|u(X^u_\tau, \tau)\big\|^2\, d\tau + g(X^u_1)\right] \quad \text{subject to Equation (2).} \tag{3}$$

Q-learning with Adjoint Matching (QAM) [24] casts off-policy RL in this framework by taking X^u_1 as the action at state s and the negative critic $-Q^\pi(s, \cdot)$ as the terminal cost g. The control u then steers the pretrained policy toward critic-preferred actions, while the quadratic term penalizes deviation from π_{base} [9]. QAM solves Equation (3) with Adjoint Matching in two steps. First, it propagates the critic gradient at the final action back along the sampling trajectory. This gives the

adjoint $\widetilde{a}_\tau$, obtained by integrating the following lean adjoint ODE backward in time, where the term in parentheses is the base drift b of Equation (1),

$$\widetilde{a}_{\tau-h} = \widetilde{a}_\tau + h\,\widetilde{a}_\tau^\top \nabla_x \left(2v^{\text{base}}(X_\tau, \tau) - \frac{1}{\tau} X_\tau \right), \qquad \widetilde{a}_1 = -\nabla_{x_1} Q^\pi(s, X_1). \tag{4}$$

Second, the fine-tuned velocity field v_θ^{ft}, whose deviation from v^{base} defines the control u, is updated by minimizing the Adjoint Matching loss

$$\mathcal{L}_{\text{Adj-Match}}(\theta) = \sum_{\tau \in \{0,\dots,1-h\}} \left\| \frac{2}{\sigma(\tau)} \big(v_\theta^{\text{ft}}(X_\tau, \tau) - v^{\text{base}}(X_\tau, \tau) \big) + \sigma(\tau)\, \widetilde{a}_\tau \right\|^2, \tag{5}$$

without backpropagating through the sampling chain. Unlike our formulation, however, QAM does not introduce λ in Equation (2), so it can be viewed as the special case of ours with $\lambda = 1$.

3 Method

In this section, we introduce *Trust Region Q Adjoint Matching* (TRQAM), a framework for stable off-policy fine-tuning of pretrained flow policies. We first identify that small critic errors can be exponentially amplified into large policy errors, often leading to performance collapse. To address this, TRQAM limits the deviation from the pretrained policy with a trust-region parameter λ, which scales the control term in the drift (Equation (2)). Concretely, we show that this deviation is bounded by the path-space KL (Proposition 1), and that λ is exactly the weight of this KL in the SOC objective by Girsanov's theorem (Theorem 1), which makes λ a natural dual variable for a KL budget. TRQAM adapts λ by dual descent throughout training to keep this KL within a prescribed budget ε_{KL}. Since this KL can be estimated at no extra cost, this trust region adds no overhead over QAM.

We develop our core contribution by investigating the following questions:

- Why does a fixed λ fail against critic errors in off-policy RL? (Section 3.1)
- How can λ limit the effect of these critic errors? (Section 3.2)
- How do we adapt λ throughout training? (Section 3.3)
- Why must λ act inside the SOC sampling process rather than through an external KL penalty? (Section 3.4)

3.1 Why does a fixed λ fail against critic errors in off-policy RL?

In off-policy RL, the learned critic is inevitably imperfect. Its approximation error compounds through bootstrapping and is especially severe under distributional shift [12, 22]. The central risk of critic-guided policy improvement is that small critic errors can induce large deviations from the policy the true critic would produce, which we call policy errors. We formalize this risk in the following lemma, which applies to any policy obtained by exponentially tilting the pretrained policy π_{base}. This class includes QAM, since the optimum of its memoryless SOC objective has this form [9, 24].

Lemma 1 (Exponential amplification of critic errors)**.** *Fix a state $s \in \mathcal{S}$. Let $Q : \mathcal{A} \to \mathbb{R}$ be the true critic and $\widetilde{Q} : \mathcal{A} \to \mathbb{R}$ a learned critic with $\|Q - \widetilde{Q}\|_\infty \le \varepsilon$. Define the policies obtained by exponentially tilting π_{base} with each critic,*

$$\pi_Q(a \mid s) \propto \pi_{\text{base}}(a \mid s)\, e^{\beta Q(a)}, \qquad \pi_{\widetilde{Q}}(a \mid s) \propto \pi_{\text{base}}(a \mid s)\, e^{\beta \widetilde{Q}(a)},$$

where $\beta > 0$ is the inverse temperature. Then the following inequalities hold:

$$D_{\text{KL}}(\pi_Q \,\|\, \pi_{\widetilde{Q}}) \le 2\beta\varepsilon, \qquad \text{TV}(\pi_Q, \pi_{\widetilde{Q}}) \le \frac{1}{2}\big(e^{2\beta\varepsilon} - 1\big).$$

Proof. See Appendix D.1 for the full proof. □

Since the total-variation bound grows exponentially in $\beta\varepsilon$, a large β amplifies critic errors into large policy errors. Lowering β avoids this amplification, but it also tilts π_{base} less toward high-value actions, so the policy improves little even when the critic is accurate. The lemma thus shows that

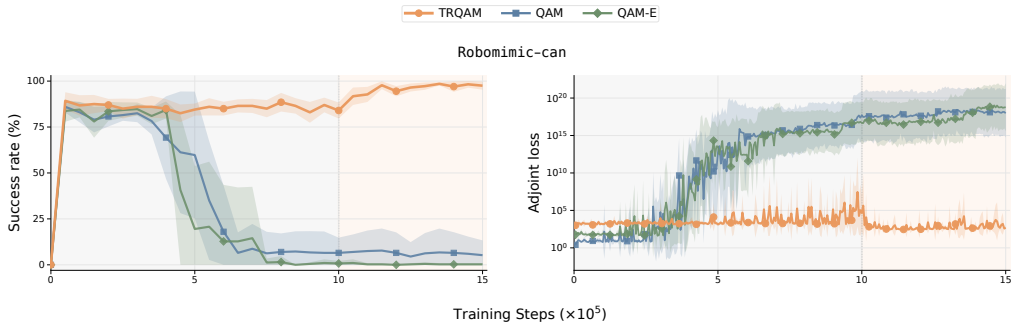


Figure 2: **Empirical fragility of fixed-temperature Adjoint Matching.** On `Robomimic-can`, the adjoint-matching loss in QAM and QAM-E grows above 10^{20} even with gradient clipping (min–max across seeds), driving task success from over 80% to near zero (± 1 standard deviation), while TRQAM remains stable. This collapse persists across most hyperparameter settings we tested on both `Robomimic-lift` and `Robomimic-can`. Appendix Figures 11 and 12 show the full sweep.

no single fixed β can both exploit a reliable critic and protect against an unreliable one. We observe this empirically on Robomimic [32], where fixed-temperature QAM and QAM-E show a diverging adjoint loss and a collapse in task success even with gradient clipping (Figure 2).

The same tension arises for λ in our SOC setting, where λ controls how strongly the policy is held close to π_{base}. Like a large β, a small λ lets the policy deviate far from π_{base}, while a large λ keeps the policy close to π_{base}. Hence no single fixed λ resolves this tension either, which motivates adapting λ throughout training. Doing so requires relating λ to β, and thereby to the critic errors of Lemma 1, which we establish next.

3.2 How λ limits critic errors through the KL

We build this relation in steps, starting from how λ enters the sampling process. In the controlled SDE in Equation (2), the total diffusion $\sqrt{\lambda}\,\sigma(\tau)$ is fixed by the memoryless schedule, so λ enters only through $\sigma(\tau) \propto 1/\sqrt{\lambda}$. Since $\sigma(\tau)$ multiplies the control u in the drift of the controlled SDE, a larger λ keeps the controlled SDE closer to the base SDE in Equation (1). We formalize this with Girsanov's theorem, showing that the quadratic control cost equals the path-space KL between the fine-tuned and base trajectory distributions, with λ as the inverse coefficient.

Theorem 1 (SOC control cost = path-space KL)**.** *Let $\mathbb{P}^u$ and $\mathbb{P}^{\text{base}}$ denote the distributions over trajectories induced by the controlled and base SDEs in Equation (2) and Equation (1), respectively. Then,*

$$D_{\text{KL}}\big(\mathbb{P}^u \,\|\, \mathbb{P}^{\text{base}}\big) \;=\; \mathbb{E}_{\mathbf{X}\sim\mathbb{P}^u}\left[\frac{1}{2\lambda}\int_0^1 \|u(X_\tau,\tau)\|^2\, d\tau\right].$$

Proof. See Appendix D.2 for the full proof. □

Theorem 1 thus rewrites the SOC objective in Equation (3) as $\lambda\, D_{\text{KL}}(\mathbb{P}^u \,\|\, \mathbb{P}^{\text{base}}) + \mathbb{E}[g(X_1^u)]$, so λ is exactly the weight of the path-space KL within the SOC objective itself. This path-space KL is itself a natural trust region for flow policies, since it constrains the whole generation trajectory rather than only the terminal action, in line with trajectory-level formulations [48, 63]. The inverse temperature β, however, acts on the terminal action distribution $\pi_\theta(\cdot \mid s)$. To connect the two, the following proposition shows that the path-space KL upper-bounds the terminal KL $D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{base}})$.

Proposition 1 (Terminal KL upper-bounded by path-space KL)**.** *Let $\mathbb{P}^u$ and $\mathbb{P}^{\text{base}}$ denote the distributions over trajectories induced by the controlled and base SDEs in Equation (2) and Equation (1), respectively, and let $\pi_\theta(\cdot \mid s)$ and $\pi_{\text{base}}(\cdot \mid s)$ denote the corresponding terminal action distributions at $\tau = 1$. Then,*

$$D_{\text{KL}}\big(\pi_\theta(\cdot \mid s) \,\|\, \pi_{\text{base}}(\cdot \mid s)\big) \;\le\; D_{\text{KL}}\big(\mathbb{P}^u \,\|\, \mathbb{P}^{\text{base}}\big).$$

Proof. See Appendix D.3 for the full proof. □

Proposition 1 bounds this terminal KL, but does not say how it depends on β. To make this dependence explicit, we use a property of exponential tilting. It is separate from Lemma 1, which concerns the KL between two tilted policies under critic error.

Proposition 2 (Tilted-policy KL is non-decreasing in β)**.** *Fix a state $s \in \mathcal{S}$ and a bounded critic $Q : \mathcal{A} \to \mathbb{R}$. For $\beta > 0$, define the exponentially tilted policy and its log-partition function*

$$\pi_\beta(a \mid s) = \frac{\pi_{\text{base}}(a \mid s)\, e^{\beta Q(a)}}{Z(\beta)}, \qquad \psi(\beta) = \log Z(\beta),$$

where $Z(\beta) = \mathbb{E}_{a\sim\pi_{\text{base}}}[e^{\beta Q(a)}]$. Then the following identities hold:

$$D_{\text{KL}}(\pi_\beta \,\|\, \pi_{\text{base}}) = \beta\psi'(\beta) - \psi(\beta), \qquad \frac{d}{d\beta} D_{\text{KL}}(\pi_\beta \,\|\, \pi_{\text{base}}) = \beta \text{Var}_{\pi_\beta}[Q] \geq 0.$$

In particular, $D_{\text{KL}}(\pi_\beta \,\|\, \pi_{\text{base}})$ is non-decreasing in β.

Proof. See Appendix D.4 for the full proof. □

Under Adjoint Matching (AM), the terminal distribution of the optimal controlled process is this tilted policy [9], so training drives π_θ toward it. The terminal KL of Proposition 1 therefore satisfies $D_{\text{KL}}(\pi_\theta \,\|\, \pi_{\text{base}}) \approx D_{\text{KL}}(\pi_\beta \,\|\, \pi_{\text{base}})$, which grows with β.

Together, these results relate λ and β through the terminal KL, as summarized below.

Remark: Connection between trust-region parameter λ and inverse temperature β

$$\lambda \uparrow \underset{\text{Thm. 1}}{\Longrightarrow} \underbrace{D_{\text{KL}}(\mathbb{P}^u \| \mathbb{P}^{\text{base}})}_{\text{path-space KL}} \downarrow \underset{\text{Prop. 1}}{\Longrightarrow} \underbrace{D_{\text{KL}}(\pi_\theta \| \pi_{\text{base}})}_{\text{terminal KL}} \downarrow$$

$$\beta \uparrow \Longrightarrow \underbrace{D_{\text{KL}}(\pi_\theta \| \pi_{\text{base}})}_{\substack{\text{terminal KL} \\ \text{(AM, Prop. 2)}}} \uparrow \quad \text{and} \quad \underbrace{\tfrac{1}{2}\left(e^{2\beta\varepsilon} - 1\right)}_{\substack{\text{critic-error amplification} \\ \text{(Lem. 1)}}} \uparrow$$

In short, λ and β move the fine-tuned policy π_θ in opposite directions relative to π_{base}. A larger λ keeps π_θ closer to π_{base}, while a larger β pushes it further away and also amplifies critic errors (Lemma 1). This confirms the claim of Section 3.1 that a fixed λ runs into the same problem as a fixed β. What λ offers in addition is an exact link to the path-space KL (Theorem 1). This KL therefore gives a measurable budget on how far the policy moves from π_{base}. Rather than fixing λ, we treat it as the dual variable of this budget and adapt it throughout training so that the path-space KL stays within a prescribed ε_{KL}. The next section makes this concrete with a tractable KL estimator and a dual update for λ.

3.3 Adaptive λ via projected dual descent

To keep the realized path-space KL within a target bound ε_{KL}, we need a tractable KL estimator and a principled rule for updating λ based on it. We use the fact that the discretized memoryless OT sampler with T steps of size $h = 1/T$ on the grid $\tau_k = kh$ and diffusion schedule $\nu(\tau) = \sqrt{2(1-\tau)/\tau}$ is a Markov chain whose Gaussian transitions share the same covariance, so per-step KL divergences admit a closed form. Summing these per-step KLs along a trajectory approximately recovers the path-space KL, which we estimate via Monte Carlo over sampled trajectories (derivation in Appendix E),

$$\widehat{D}_n = \mathbb{E}_{\mathbf{X}\sim\mathbb{P}^u}\left[\sum_{k=0}^{T-1} \frac{2h}{\nu(\tau_k)^2} \left\| v_\theta^{\text{ft}}(X_{\tau_k}, \tau_k) - v^{\text{base}}(X_{\tau_k}, \tau_k) \right\|^2\right]. \tag{6}$$

This estimate comes at no extra cost, since the velocity difference $v_\theta^{\text{ft}} - v^{\text{base}}$ is already computed for the Adjoint Matching loss in Equation (5). To reduce variance, we smooth $\widehat{D}_n$ with an exponential moving average: $\overline{D}_n \leftarrow (1-\rho_D)\,\overline{D}_{n-1} + \rho_D\,\widehat{D}_n$. Given this estimator, we adapt λ by interpreting it as the dual variable of a KL-constrained improvement problem $\max_u \mathbb{E}[Q^\pi(X_1^u)]$ subject to

Algorithm 1 QAM

Require: v^{base}, Q_ϕ^π, training steps N

$v_\theta^{\text{ft}} \leftarrow v^{\text{base}}$
for $n = 0, \dots, N-1$ **do**
  TD update of Q_ϕ^π
  Sample a trajectory by v_θ^{ft}
  Solve the lean adjoint ODE in Equation (4)
  Update θ by Equation (5)
end for

Algorithm 2 TRQAM (ours)

Require: v^{base}, Q_ϕ^π, KL budget ε_{KL}, training steps N

$v_\theta^{\text{ft}} \leftarrow v^{\text{base}}$, $\lambda \leftarrow \lambda_0$
for $n = 0, \dots, N-1$ **do**
  TD update of Q_ϕ^π
  Sample a trajectory by v_θ^{ft}
  Solve the lean adjoint ODE in Equation (4)
  Update θ by Equation (5) with λ_n
  Estimate the path-space KL by Equation (6)
  Dual update of λ_n by Equation (7)
end for

Figure 3: Simplified versions of the QAM and TRQAM algorithms. The parts TRQAM adds are marked in blue, and Appendix A gives the full algorithm.

$D_{\text{KL}}(\mathbb{P}^u \| \mathbb{P}^{\text{base}}) \le \varepsilon_{\text{KL}}$, and apply projected dual descent with a fixed step size $\eta_\lambda > 0$ (derivation in Appendix F),

$$\lambda_{n+1} \;\leftarrow\; \max\{0,\; \lambda_n + \eta_\lambda\,(\overline{D}_n - \varepsilon_{\text{KL}})\}. \tag{7}$$

Intuitively, when the realized KL exceeds the bound, λ rises and the controlled SDE becomes more conservative. When it falls below, λ decreases and allows more aggressive improvement.

Algorithms 1 and 2 compare QAM and TRQAM side by side. TRQAM keeps every step of QAM and adds three changes, highlighted in blue. It uses the current λ_n in the Adjoint Matching loss through $\sigma(\tau)$, estimates the path-space KL from the velocity difference already computed in the Adjoint Matching loss, and updates λ by dual descent. Appendix A gives the full algorithm.

3.4 Internal vs. external KL regularization

The dual update in Equation (7) can be combined with the SOC objective in two ways,

$$\underbrace{\min_\theta \; \mathcal{L}_{\text{Adj-Match}}(\theta) + \lambda \cdot \overline{D}_n(\theta)}_{\text{External: } \lambda \text{ is a regularization weight}} \quad \text{vs.} \quad \underbrace{\min_\theta \; \mathcal{L}_{\text{Adj-Match}}(\theta) \;\; \text{with } \lambda \text{ in Equation (2)}}_{\text{Internal (TRQAM): } \lambda \text{ appears in the SOC sampling process}}.$$

In the internal form (right), λ acts only on the control term $\sigma(\tau)\,u$ of the controlled SDE, as shown in Section 3.2. Increasing λ therefore pulls the controlled SDE toward the base SDE, and Theorem 1 ties the path-space KL to λ by an exact identity. The dual update thus enforces the trust region through the sampling process itself. However, the external form (left) uses λ only to weight a loss penalty, which must compete with the Adjoint Matching loss during optimization. Under strong critic guidance, the critic term can dominate this penalty, and the realized KL drifts far from ε_{KL}. Table 7 in Appendix G summarizes these differences, and Section 4.2 validates them empirically.

4 Experiments

We evaluate TRQAM on off-policy fine-tuning of pretrained flow policies in the offline-to-online setting. We use OGBench [40] (50 tasks) for main comparison, and Robomimic [32] for ablation and mechanism studies. All methods share the same pretrained flow policy and training schedule.

Setup. OGBench [40] is an offline goal-conditioned RL benchmark spanning 10 domains, from which we evaluate on 50 tasks. While OGBench is originally designed for offline goal-conditioned RL, we use its reward-based single-task variants. Robomimic [32] is a demonstration-based manipulation benchmark used to test the stability. For all manipulation tasks (*e.g.*, OGBench's `scene`, `cube`, `puzzle` domains and all Robomimic tasks), we use action-chunked policies with chunk size 5 [25].

On OGBench, we compare our method against six off-policy fine-tuning baselines, namely FQL [41], CGQL-Linex [7], DSRL [56], IFQL [14], QAM, and QAM-E [24]. On Robomimic, we focus on the adjoint-matching variants (*i.e.*, QAM and QAM-E), and DSRL as a non-adjoint reference. We refer to Appendix B for details on each baseline. Since trust-region fine-tuning regulates deviation

Table 1: **Offline RL on 50 OGBench [40] tasks at 1M training steps (8 seeds).** Mean success rate (%) with ±1 standard deviation. Per-task results for all 50 tasks are in Table 16.

		al	ag	hm	hl	scene	p33	p44	c2	c3	c4	all
		5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	5 tasks	50 tasks
BACKPROP	FQL	38 ±9	2 ±6	74 ±5	2 ±1	70 ±5	25 ±10	9 ±7	44 ±4	7 ±5	9 ±5	28
GUIDANCE	CGQL-L	48 ±7	7 ±5	57 ±2	6 ±3	58 ±1	0 ±0	0 ±0	55 ±2	0 ±1	1 ±1	23
POST PROCESSING	DSRL	53 ±2	1 ±1	53 ±10	1 ±1	80 ±0	100 ±0	61 ±8	72 ±4	34 ±6	9 ±3	46
	IFQL	29 ±8	12 ±3	93 ±2	30 ±7	36 ±1	64 ±4	42 ±4	9 ±2	24 ±7	6 ±3	35
ADJOINT MATCHING	QAM	62 ±9	29 ±4	64 ±7	4 ±3	64 ±4	15 ±3	1 ±1	71 ±2	19 ±6	18 ±3	35
	QAM-E	86 ±3	6 ±8	60 ±6	4 ±5	63 ±6	89 ±4	54 ±8	71 ±3	11 ±4	9 ±3	45
OURS	TRQAM	**89** ±4	**41** ±4	84 ±3	**36** ±4	**79** ±1	**100** ±0	**99** ±1	**81** ±3	**50** ±5	**19** ±5	**68**

from a pretrained prior, this prior must itself encode meaningful behavior. Therefore, all methods are pretrained for 300K steps with behavior cloning, and are then fine-tuned for 10^6 offline steps followed by 0.5×10^6 online steps. We report average success rate (%) over 8 seeds, and the detailed hyperparameters are in Appendix C.3.

Abbreviations. We evaluate on 10 OGBench domains and abbreviate their names in tables and figures for compactness: `puzzle-4x4` (`p44`), `cube-double` (`c2`), `cube-triple` (`c3`), `cube-quadruple` (`c4`), `scene`, `humanoidmaze-medium` (`hm`), `humanoidmaze-large` (`hl`), `antmaze-large` (`al`), `antmaze-giant` (`ag`), and `puzzle-3x3` (`p33`).

4.1 Main results on offline and offline-to-online RL

Table 1 reports offline success rates after 1M training steps. TRQAM achieves 68% aggregate success across 50 tasks, improving on QAM (35%) by 33 points, on its strongest variant QAM-E (45%) by 23 points, and on the strongest non-adjoint baseline DSRL (46%) by 22 points, with the largest gains on long-horizon and combinatorial domains. This lead is sustained through the offline-to-online transition. The per-task curves in Figures 18 and 19 show that TRQAM remains the strongest method through 500K steps of online fine-tuning.

A natural explanation for TRQAM's gains is that it leverages the pretrained policy more effectively. Figure 4 tests this by running TRQAM, QAM-E, and QAM both from a pretrained flow policy (dashed) and from scratch (solid) under an identical offline-to-online protocol. The contrast is clear. TRQAM benefits substantially from the pretrained prior and reaches high success much earlier than its scratch counterpart. In contrast, QAM and QAM-E show little to no benefit from the same pretrained initialization, and their pretrained and scratch curves remain close throughout training.

4.2 What mechanism drives its gains?

To isolate what drives the asymmetry above, we compare three variants differing only in how λ is regulated. QAM keeps λ constant in our framework. QAM + External KL adapts λ through a conventional KL regularization loss. TRQAM adapts λ inside the SOC sampling process. These variants isolate two design axes, namely *adaptation* (constant KL vs. adaptive KL) and *internalization* (external loss penalty vs. internalized SOC sampling process).

1. Adaptive KL outperforms constant KL. Both adaptive KL variants, TRQAM and QAM with external KL regularization, substantially outperform QAM on `cube-triple-task1` and `humanoidmaze-medium-task1` (Figure 5). The same asymmetry appears on Robomimic, where QAM exhibits the diverging adjoint loss and collapsing task success of Figure 2 while both adaptive variants remain stable (Appendix H.10). These results are consistent with Lemma 1's exponential amplification of critic errors under fixed temperature.

2. Internal KL outperforms external KL regularization. Among the two adaptive variants, only TRQAM enforces the prescribed KL budget. On `Robomimic-lift` and `Robomimic-can` with $\varepsilon_{\mathrm{KL}} = 0.1$, TRQAM tightly tracks the bound throughout training, while QAM + External KL

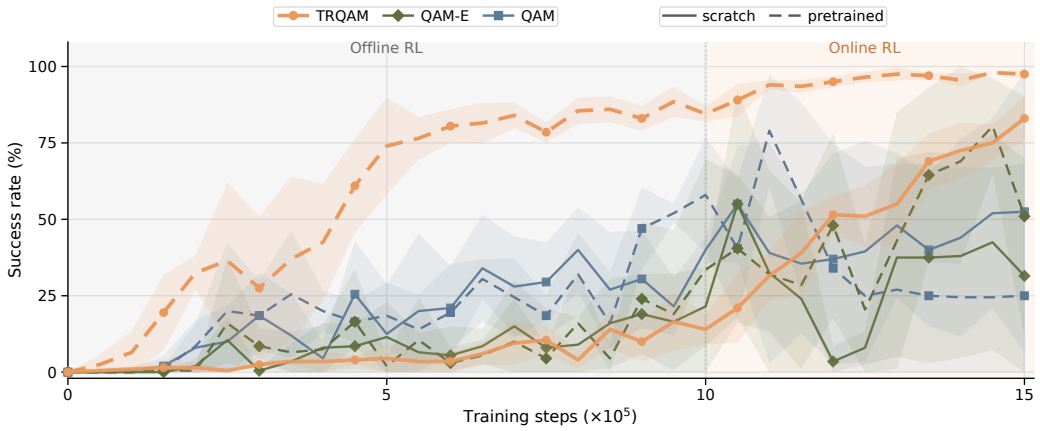


Figure 4: **Pretraining alone is not sufficient.** On `humanoidmaze-medium-task1`, each algorithm is run from a pretrained flow policy (dashed) and from scratch (solid). TRQAM benefits substantially from the pretrained prior, while QAM and QAM-E show little to no benefit from the same pretrained initialization, with their pretrained and scratch curves remaining close throughout training. Shaded regions denote standard deviation across seeds.

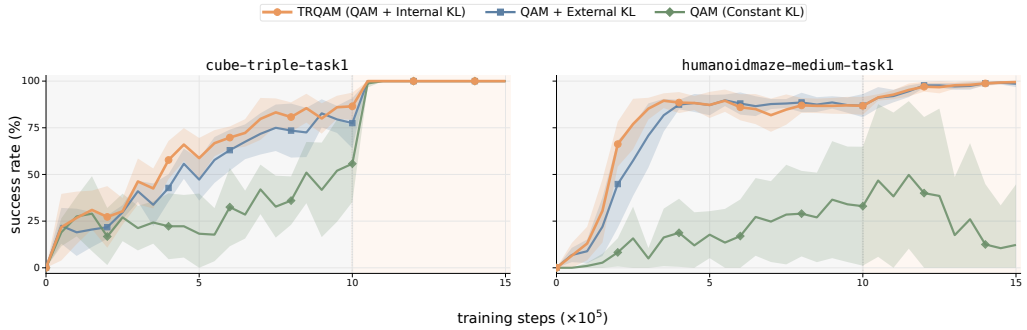


Figure 5: **Adaptation is necessary.** Both adaptive variants (TRQAM and QAM + External KL) outperform QAM (constant λ) on `cube-triple-task1` and `humanoidmaze-medium-task1`, consistent with Lemma 1. A fixed temperature can amplify critic errors into policy deviations, while adaptation can mitigate this. Shaded regions denote standard deviation across seeds.

lets the realized KL drift above $\varepsilon_{\mathrm{KL}}$ with corresponding success-rate degradation (Figure 6). By Theorem 1, only the internal parameterization makes λ the exact weight of the path-space KL in the SOC objective, whereas an external penalty enters only as an additive loss term that strong critic guidance can override. The same pattern holds across all three Robomimic tasks and six KL budgets (Figures 15 to 17).

4.3 Sensitivity Analysis

Since $\varepsilon_{\mathrm{KL}}$ directly sets how far TRQAM can move from the pretrained policy, it is the main hyperparameter of our method. To measure its effect, we sweep $\varepsilon_{\mathrm{KL}}$ from 0.5 to 4 in steps of 0.5 on four representative OGBench tasks (`humanoidmaze-medium-task1`, `humanoidmaze-large-task1`, `cube-double-task2`, `cube-triple-task2`), with the same training setup as Section 4.1. Figure 7 shows two patterns. First, success rate changes smoothly with $\varepsilon_{\mathrm{KL}}$ on every task. Second, tight budgets are best across all four tasks. The similar pattern holds across most of the 10 OGBench domains in Appendix H.3, with `puzzle-4x4` as the exception where larger budgets monotonically improve performance, in keeping with its notably larger state space. Because TRQAM tightly enforces the chosen budget (Section 4.2), tuning $\varepsilon_{\mathrm{KL}}$ to task structure produces predictable, controlled behavior, so adapting TRQAM to a new environment comes down to choosing $\varepsilon_{\mathrm{KL}}$ based on task structure.

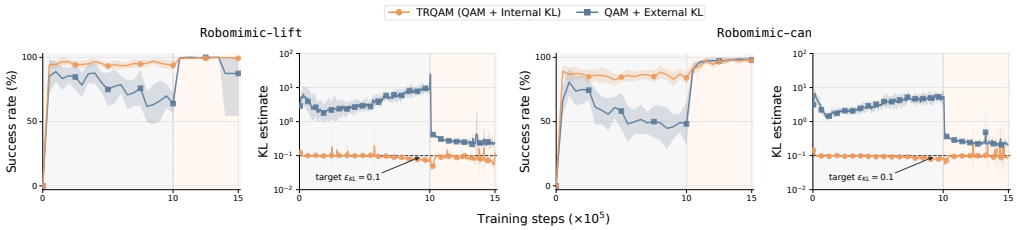


Figure 6: **Internalization is necessary.** On `Robomimic-lift` & `Robomimic-can` with $\varepsilon_{\mathrm{KL}} = 0.1$, TRQAM tightly tracks the target KL bound throughout training, while QAM + External KL lets the realized KL drift well above it, with corresponding success rate degradation. By Theorem 1, only the internal parameterization (TRQAM) makes λ the exact weight of the path-space KL in the SOC objective, while the external loss penalty (QAM + External KL) can be overridden by strong critic guidance. Shaded regions denote standard deviation across seeds.

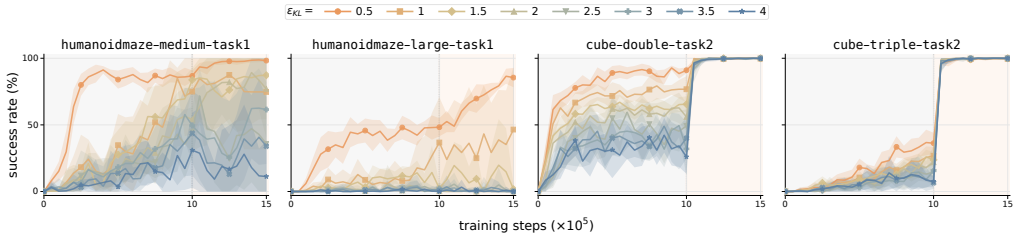


Figure 7: **Sensitivity Analysis.** Success-rate curves on four OGBench tasks under varying KL budgets. Two patterns emerge. First, success rate changes smoothly with $\varepsilon_{\mathrm{KL}}$ on every task, making the budget a predictable knob. Second, tight budgets are best on all four tasks. Domains with much larger state spaces, such as `puzzle-4x4`, instead benefit from larger budgets (Figure 10).

5 Related Work

Offline-to-online RL. Pretraining a policy and value function on offline data and then fine-tuning online is a standard recipe for sample-efficient RL [3, 15, 16, 21–23, 30, 34, 35, 47, 55]. A central challenge is the distribution shift at the transition, which destabilizes the value function and induces catastrophic forgetting [3, 35, 58], motivating trust-region-style constraints during fine-tuning.

Fine-tuning flow and diffusion policies. Flow matching and diffusion policies [6, 26] parameterize multi-modal action distributions and are increasingly pretrained at scale [37, 4, 44, 45]. A growing body of work trains such policies with RL [5, 10, 14, 28, 38, 41, 48, 56, 57, 60, 61, 63, 64, 66], each navigating a tradeoff between policy expressivity, computational cost, and training stability.

KL trust regions in RL. KL regularization stabilizes policy updates as a hard constraint or soft penalty [1, 43, 50, 51, 59], sometimes with strength adapted via dual updates (e.g., SAC [13] for entropy). These approaches enforce the constraint via auxiliary losses external to the policy.

6 Conclusion

We introduce *Trust Region Q Adjoint Matching* (TRQAM), a stable off-policy fine-tuning method for pretrained flow policies. TRQAM adapts a trust-region parameter λ inside the SOC sampling process. Letting λ scale the control term in the drift makes λ the exact weight of the path-space KL in the SOC objective (Theorem 1), so dual descent on λ enforces the target bound within the SOC formulation rather than through a loss-level penalty. Across 50 OGBench tasks, TRQAM improves the strongest baseline by 22 points, with the largest gains on long-horizon and combinatorial domains. On Robomimic, it remains stable where fixed-temperature Adjoint Matching collapses. PPO became a standard tool for on-policy RL largely because it made policy updates stable [51]. We hope TRQAM plays a similar role for off-policy fine-tuning of pretrained flow policies, where a prescribed KL budget keeps each update close to a trusted prior. As pretrained flow policies grow into large vision-language-action models, we believe such stability will matter even more. We hope this work encourages future research to treat stability as a central goal when fine-tuning these models with RL.

Acknowledgments and Disclosure of Funding

This work was supported by Institute for Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (RS-2022-II220953, Self-directed AI Agents with Problem-solving Capability; RS-2019-II190075, Artificial Intelligence Graduate School Program (KAIST)). We thank RLWRLD Inc. for providing compute resources to conduct experiments in this work.

YH thanks the anonymous reviewers for their constructive feedback, and Sihyun Yu, Taeyoung Kim, Seungku Kim, Daewon Choi, Woomin Song, Junbeom Kim, Beomjun Kim, Myungkyu Koo, Suhyeok Jang, Huiwon Jang, Sanghyeok Lee, John Won, Dongyoung Kim, and Jimin Lee, for their helpful feedback and insightful discussions.

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *International Conference on Learning Representations*, 2018.

[2] Michael S Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. In *Journal of Machine Learning Research*, 2025.

[3] Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. Efficient online reinforcement learning with offline data. In *International Conference on Machine Learning*, 2023.

[4] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems*, 2025.

[5] Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Xiang Li, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, Tiejun Huang, Yu Wang, and Chao Yu. π_{RL}: Online RL fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 2026.

[6] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, 2021.

[8] Carles Domingo-Enrich, Jiequn Han, Brandon Amos, Joan Bruna, and Ricky Tian Qi Chen. Stochastic optimal control matching. In *Advances in Neural Information Processing Systems*, 2024.

[9] Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky Tian Qi Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. In *International Conference on Learning Representations*, 2025.

[10] Perry Dong, Qiyang Li, Dorsa Sadigh, and Chelsea Finn. EXPO: Stable reinforcement learning with expressive policies. In *International Conference on Learning Representations*, 2026.

[11] JaeHyeok Doo, Byeongguk Jeon, Seonghyeon Ye, Kimin Lee, and Minjoon Seo. Q-Flow: Stable and expressive reinforcement learning with flow-based policy. In *International Conference on Machine Learning*, 2026.

[12] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, 2018.

[14] 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.

[15] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep Q-learning from demonstrations. In *AAAI Conference on Artificial Intelligence*, 2018.

[16] Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. Imitation bootstrapped reinforcement learning. In *International Conference on Learning Representations*, 2024.

[17] Chia-Yu Hung, Navonil Majumder, Haoyuan Deng, Liu Renhang, Yankang Ang, Amir Zadeh, Chuan Li, Dorien Herremans, Ziwei Wang, and Soujanya Poria. NORA-1.5: A vision-language-action model trained using world model- and action-based preference rewards. *arXiv preprint arXiv:2511.14659*, 2025.

[18] Tao Jiang, Tianyuan Yuan, Yicheng Liu, Chenhao Lu, Jianning Cui, Xiao Liu, Shuiqi Cheng, Jiyang Gao, Huazhe Xu, and Hang Zhao. Galaxea open-world dataset and G0 dual-system VLA model. *arXiv preprint arXiv:2509.00576*, 2025.

[19] Changyeon Kim, Haeone Lee, Younggyo Seo, Kimin Lee, and Yuke Zhu. DEAS: DEtached value learning with action sequence for scalable offline RL. In *International Conference on Learning Representations*, 2026.

[20] Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, Beomjun Kim, Byungjun Yoon, Changsung Jang, Daewon Choi, Dongsu Han, et al. RLDX-1 technical report. *arXiv preprint arXiv:2605.03269*, 2026.

[21] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit Q-Learning. In *International Conference on Learning Representations*, 2022.

[22] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2020.

[23] Kun Lei, Zhengmao He, Chenhao Lu, Kaizhe Hu, Yang Gao, and Huazhe Xu. Uni-O4: Unifying online and offline deep reinforcement learning with multi-step on-policy optimization. In *International Conference on Learning Representations*, 2024.

[24] Qiyang Li and Sergey Levine. Q-learning with adjoint matching. In *International Conference on Learning Representations*, 2026.

[25] Qiyang Li, Zhiyuan Zhou, and Sergey Levine. Reinforcement learning with action chunking. In *Advances in Neural Information Processing Systems*, 2025.

[26] Yaron Lipman, Ricky Tian Qi Chen Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

[27] Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky Tian Qi Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

[28] Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *Advances in Neural Information Processing Systems*, 2025.

[29] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations*, 2023.

[30] Jianlan Luo, Zheyuan Hu, Charles Xu, You Liang Tan, Jacob Berg, Archit Sharma, Stefan Schaal, Chelsea Finn, Abhishek Gupta, and Sergey Levine. SERL: A software suite for sample-efficient robotic reinforcement learning. In *IEEE International Conference on Robotics and Automation*, 2024.

[31] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, 2024.

[32] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, 2021.

[33] Vivek Myers, Bill Zheng, Benjamin Eysenbach, and Sergey Levine. Offline goal-conditioned reinforcement learning with quasimetric representations. In *Advances in Neural Information Processing Systems*, 2025.

[34] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. AWAC: Accelerating online reinforcement learning with offline datasets. In *International Conference on Learning Representations*, 2021.

[35] Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-QL: Calibrated offline RL pre-training for efficient online fine-tuning. In *Advances in Neural Information Processing Systems*, 2023.

[36] Nikolas Nüsken and Lorenz Richter. Solving high-dimensional Hamilton–Jacobi–Bellman PDEs using neural networks: perspectives from the theory of controlled diffusions and measures on path space. *Partial differential equations and applications*, 2:1–48, 2021.

[37] NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Jim Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[38] Aditya Oberai, Seohong Park, and Sergey Levine. Reversal Q-Learning. *arXiv preprint arXiv:2606.17551*, 2026.

[39] Bernt Øksendal. *Stochastic differential equations*. Springer, 2003.

[40] Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. OGBench: Benchmarking offline goal-conditioned RL. In *International Conference on Learning Representations*, 2025.

[41] Seohong Park, Qiyang Li, and Sergey Levine. Flow Q-Learning. In *International Conference on Machine Learning*, 2025.

[42] Ahmad Parsian and SNUA Kirmani. Estimation under LINEX loss function. In *Handbook of applied econometrics and statistical inference*, pages 75–98. CRC Press, 2002.

[43] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. In *International Conference on Learning Representations*, 2021.

[44] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi,

Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: a VLA that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025.

[45] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Conference on Robot Learning*, 2025.

[46] Yury Polyanskiy and Yihong Wu. *Information Theory: From Coding to Learning*. Cambridge University Press, 2025.

[47] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems*, 2018.

[48] Allen Z. Ren, Justin Lidard, Lars L. Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, 2025.

[49] Moritz Reuss, Hongyi Zhou, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Otto, and Rudolf Lioutikov. FLOWER: Democratizing generalist robot policies with efficient vision-language-flow models. In *Conference on Robot Learning*, 2025.

[50] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization. In *International Conference on Machine Learning*, 2015.

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

[52] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

[53] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[54] Serge Thilges, Onur Celik, Denis Blessing, Emiliyan Gospodinov, and Gerhard Neumann. Scalable maximum entropy reinforcement learning for diffusion policies via adjoint matching. *arXiv preprint arXiv:2606.22630*, 2026.

[55] Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv preprint arXiv:1707.08817*, 2018.

[56] Andrew Wagenmaker, Mitsuhiko Nakamoto, Yunchu Zhang, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Conference on Robot Learning*, 2025.

[57] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations*, 2023.

[58] Maciej Wołczyk, Bartłomiej Cupiał, Mateusz Ostaszewski, Michał Bortkiewicz, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Fine-tuning reinforcement learning models is secretly a forgetting mitigation problem. In *International Conference on Machine Learning*, 2024.

[59] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. In *International Conference on Learning Representations*, 2020.

[60] Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Jim Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual RL. In *International Conference on Learning Representations*, 2026.

[61] Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. RL token: Bootstrapping online RL with vision-language-action models. *arXiv preprint arXiv:2604.23073*, 2026.

[62] Andy Zhai, Brae Liu, Bruno Fang, Chalse Cai, Ellie Ma, Ethan Yin, Hao Wang, Hugo Zhou, James Wang, Lights Shi, Lucy Liang, Make Wang, Qian Wang, Roy Gan, Ryan Yu, Shalfun Li, Starrick Liu, Sylas Chen, Vincent Chen, and Zach Xu. Igniting VLMs toward the embodied space. *arXiv preprint arXiv:2509.11766*, 2025.

[63] Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. ReinFlow: Fine-tuning flow matching policy with online reinforcement learning. In *Advances in Neural Information Processing Systems*, 2025.

[64] Yixian Zhang, Shu'ang Yu, Tonghe Zhang, Mo Guang, Haojia Hui, Kaiwen Long, Yu Wang, Chao Yu, and Wenbo Ding. SAC flow: Sample-efficient reinforcement learning of flow-based policies via velocity-reparameterized sequential modeling. In *International Conference on Learning Representations*, 2026.

[65] Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, Tai Wang, Ya-Qin Zhang, Jingjing Liu, and Xianyuan Zhan. X-VLA: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *International Conference on Learning Representations*, 2026.

[66] Zhiyuan Zhou, Andy Peng, Charles Xu, Qiyang Li, Tobias Springenberg, Kevin Frans, and Sergey Levine. Test-time gradient guidance of flow policies in reinforcement learning. *arXiv preprint arXiv:2606.11087*, 2026.

Contents

A Algorithm **17**

B Baselines **17**

C Experimental details **19**

C.1 Domains and tasks 19

C.2 Compute resources 19

C.3 Hyperparameters 20

D Proofs **22**

D.1 Proof of Lemma 1 (exponential amplification of critic errors) 22

D.2 Proof of Theorem 1 (path-space KL identity) 23

D.3 Proof of Proposition 1 (terminal KL upper-bounded by path-space KL) 24

D.4 Proof of Proposition 2 (tilted-policy KL is non-decreasing in β) 24

E Path-space KL surrogate under OT memoryless discretization **24**

F Primal–dual derivation of KL-budgeted improvement **25**

G Detailed comparison of internal and external KL regularization **26**

H Additional experiments **27**

H.1 KL-budget tracking 27

H.2 Time-varying KL budget 28

H.3 Sensitivity analysis 28

H.4 A single global KL budget 28

H.5 Gap between the path-space KL and the terminal KL 30

H.6 Controlling the deviation with λ versus β 30

H.7 Robustness to the dual-update hyperparameters 31

H.8 Number of critics 31

H.9 Computational cost 31

H.10 Stability stress test on Robomimic 32

I Full OGBench results **32**

J Broader impacts **41**

K Limitations and future work **41**

A Algorithm

Algorithm 3 Trust Region Q Adjoint Matching (TRQAM) for fine-tuning Flow Matching policies. Blue marks TRQAM additions over QAM [24].

Input: replay buffer $\mathcal{D}$; v^{base}: pretrained (behavior) velocity field; v_θ^{ft}: fine-tuned velocity field; Q_ϕ: critic function; step size h; KL budget ε_{KL}; dual stepsize η_λ; EMA coefficient ρ_D; fine-tuning iterations N.
Initialize: $v_\theta^{\text{ft}} \leftarrow v^{\text{base}}$ with parameters θ; $\lambda_0 > 0$; $\overline{D}_0 \leftarrow 0$.
Memoryless SDE: $\sqrt{\lambda_n}\,\sigma_n(\tau) = \nu(\tau)$ for all n, where $\nu(\tau) := \sqrt{2(1-\tau)/\tau}$
for $n \in \{0, \dots, N-1\}$ **do**
Sample a batch $\mathcal{B} = \{(s_i, a_i, r_i, s_i')\}$ from $\mathcal{D}$

Critic update: Optimize ϕ w.r.t.

$$\mathcal{L}(\phi) = \frac{1}{|\mathcal{B}|} \sum_{(s,a,r,s') \in \mathcal{B}} \left[Q_\phi(s,a) - r - \gamma\, Q_{\bar{\phi}}(s',\, a' \sim \pi_\theta(\cdot \mid s')) \right]^2 \quad \triangleright \text{ TD backup}$$

Policy update: For each state $s \in \mathcal{B}$, sample a trajectory $\boldsymbol{X} = (X_\tau)_{\tau \in \{0,h,\dots,1\}}$ via the memoryless Euler scheme:

$$X_{\tau+h} = X_\tau + h\left(2v_\theta^{\text{ft}}(s, X_\tau, \tau) - \frac{1}{\tau} X_\tau\right) + \sqrt{h}\, \underbrace{\sqrt{\lambda_n}\,\sigma_n(\tau)}_{=\nu(\tau)}\, \varepsilon_\tau, \quad \varepsilon_\tau, X_0 \sim \mathcal{N}(0, I).$$

Compute the critic's action gradient: $\tilde{a}_1 \leftarrow -\nabla_{X_1} Q_\phi(s, X_1)$.
Solve the lean adjoint ODE backwards:

$$\tilde{a}_{\tau-h} = \tilde{a}_\tau + h\, \tilde{a}_\tau^\top \nabla_{X_\tau} \left(2v^{\text{base}}(s, X_\tau, \tau) - \frac{1}{\tau} X_\tau\right).$$

Stop gradient: $X_\tau \leftarrow \texttt{stopgrad}(X_\tau)$, $\tilde{a}_\tau \leftarrow \texttt{stopgrad}(\tilde{a}_\tau)$.
Optimize θ w.r.t. the Adjoint Matching objective:

$$\mathcal{L}_{\text{Adj-Match}}(\theta) = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} \sum_{\tau} \left\| \tfrac{2}{\sigma_n(\tau)} \left(v_\theta^{\text{ft}}(s, X_\tau, \tau) - v^{\text{base}}(s, X_\tau, \tau)\right) + \sigma_n(\tau)\, \tilde{a}_\tau \right\|^2$$

Trust region update:
Estimate path-space KL surrogate:

$$\widehat{D}_n = \frac{1}{|\mathcal{B}|} \sum_{s \in \mathcal{B}} \sum_{\tau} \tfrac{2h}{\nu(\tau)^2} \left\| v_\theta^{\text{ft}}(s, X_\tau, \tau) - v^{\text{base}}(s, X_\tau, \tau) \right\|^2$$

EMA smoothing: $\overline{D}_n \leftarrow (1 - \rho_D)\, \overline{D}_{n-1} + \rho_D\, \widehat{D}_n$.
Dual descent: $\lambda_{n+1} \leftarrow \max\{0,\ \lambda_n + \eta_\lambda(\overline{D}_n - \varepsilon_{\text{KL}})\}$.
end for
Output: fine-tuned velocity field v_θ^{ft}, critic Q_ϕ.

B Baselines

We compare against six baselines spanning distinct fine-tuning paradigms for flow-matching policies. Throughout, (s, a, r, s') are sampled uniformly from the replay buffer D without re-weighting. The buffer D contains the offline dataset during offline training and is augmented with online rollouts during online fine-tuning.

FQL [41]. FQL distills a multi-step flow policy into a one-step policy to avoid backpropagation through time. Conditioned on state s, the behavior-cloning rollout is given by the ODE

$$dX_\tau = v_\theta(X_\tau, \tau; s)d\tau, \quad X_0 \sim \mathcal{N}(0, I), \quad \tau \in [0, 1],$$

and we define $\mathrm{ODE}(v_\theta, s, X_0) := X_1$ as its terminal value. The one-step policy $\pi_\omega(s, X_0)$ is trained jointly with v_θ to maximize the critic while staying close to this rollout by minimizing

$$\mathcal{L}_{\text{onestep}}(\omega) = \mathbb{E}_{X_0 \sim \mathcal{N}}\Big[\underbrace{-Q(s, \pi_\omega(s, X_0))}_{\text{RL maximization}} + \underbrace{\alpha \, \|\pi_\omega(s, X_0) - \mathrm{ODE}(v_\theta, s, X_0)\|_2^2}_{\text{BC distillation}}\Big],$$

where α controls how closely π_ω stays to the BC rollout. The environment policy is π_ω.

CGQL-Linex. CGQL-Linex is a baseline introduced in Li and Levine [24], combining a BC velocity field with classifier guidance [7] from a Q-function. It trains an auxiliary intermediate critic $Q_\psi(s, X_\tau, \tau)$ on intermediate noisy actions X_τ, used to construct a guidance velocity that steers sampling toward the entropy-regularized optimal policy $\pi^\star(\cdot \mid s) \propto e^{\beta Q_\phi(s,\cdot)}$. The guidance velocity is

$$\hat{v}_\psi(X_\tau, \tau; s) := \frac{(1-\tau)\beta \, \nabla_{X_\tau} Q_\psi(s, X_\tau, \tau) + X_\tau}{\tau},$$

where $X_\tau = (1-\tau)X_0 + \tau X_1$ with $X_0 \sim \mathcal{N}(0, I)$. The final sampling velocity is $v = v^{\text{base}} + w \, \hat{v}_\psi$, where w modulates guidance strength. The intermediate critic Q_ψ is trained by minimizing a Linex regression loss [42, 33]

$$\mathcal{L}_{\text{Linex}}(\psi) = \mathbb{E}_{\tau, X_0}\left[\exp\big(\beta(Q_\phi(s, X_1) - Q_\psi(s, X_\tau, \tau))\big) + \beta Q_\psi(s, X_\tau, \tau)\right],$$

while the standard critic Q_ϕ is trained via standard TD with target actions sampled from the full velocity v, by minimizing

$$\mathcal{L}_{\text{TD}}(\phi) = \mathbb{E}\Big[\big(Q_\phi(s, a) - r - \gamma \, Q_{\bar{\phi}}(s', \mathrm{ODE}(v, s', X_0))\big)^2\Big].$$

We follow Li and Levine [24] in using a Huber-style stabilization of the Linex loss to prevent exponential blow-up.

DSRL [56]. DSRL performs RL directly in the noise space of a frozen flow policy. It trains a one-step Gaussian noise-space policy $\pi_\omega(X_0 \mid s)$ via SAC, maximizing a noise-space critic $Q_\psi(s, X_0)$ that regresses to the original action-space critic by minimizing

$$L(\psi) = \mathbb{E}_{X_0 \sim \mathcal{N}}\Big[\big(Q_\psi(s, X_0) - Q_\phi(s, \mathrm{ODE}(v_{\bar{\theta}}, s, X_0))\big)^2\Big].$$

At inference, actions are obtained by sampling $X_0 \sim \pi_\omega(\cdot \mid s)$ and pushing it through the flow policy, which gives $a = \mathrm{ODE}(v_\theta, s, X_0)$. Following Li and Levine [24], we modify the original DSRL to also fine-tune the BC velocity online (using a target network $v_{\bar{\theta}}$ for stability), which yields stronger offline-to-online performance.

IFQL. IFQL is the flow counterpart of implicit diffusion Q-learning [14], considered as a baseline in Park et al. [41]. Value learning uses IQL-style expectile regression [21]. A value network V_ξ is trained to fit an upper expectile of the critic, and the critic is bootstrapped through V_ξ, with the losses

$$\mathcal{L}_V(\xi) = \mathbb{E}\big[L_2^\kappa\big(Q_{\bar{\phi}}(s, a) - V_\xi(s)\big)\big],$$
$$\mathcal{L}_Q(\phi) = \mathbb{E}\Big[\big(r + \gamma V_\xi(s') - Q_\phi(s, a)\big)^2\Big],$$

where $L_2^\kappa(u) = |\kappa - \mathbf{1}(u < 0)| \, u^2$ is the expectile loss with parameter $\kappa \in (0.5, 1)$. Policy extraction uses rejection sampling, where N candidate actions are drawn from a BC flow policy by sampling $X_0^{(i)} \sim \mathcal{N}(0, I)$ and computing $a^{(i)} = \mathrm{ODE}(v_\theta, s, X_0^{(i)})$ for $i = 1, \ldots, N$, and the action with the highest Q_ϕ value is selected.

QAM and QAM-E [24]. QAM is the closest prior work to TRQAM. It solves a memoryless SOC problem analogous to Equation (3) but *without* the parameter λ in the controlled SDE (i.e., the special case $\lambda = 1$ of our framework), with an inverse temperature β in the terminal cost $g = -\beta\, Q_\phi(s, X_1)$. The fine-tuned velocity v_θ is trained against the BC velocity v^{base} through the Adjoint Matching loss

$$L_{\text{AM}}(\theta) = \mathbb{E}\left[\int_0^1 \left\| \frac{2\big(v_\theta(X_\tau, \tau; s) - v^{\text{base}}(X_\tau, \tau; s)\big)}{\sigma(\tau)} + \sigma(\tau)\, \tilde{a}_\tau \right\|_2^2 d\tau \right],$$

where $\tilde{a}_\tau$ is the lean adjoint state with terminal condition $\tilde{a}_1 = -\beta\, \nabla_{X_1} Q_\phi(s, X_1)$, and X_τ follows the memoryless SDE in Equation (2) with $\lambda = 1$. Element-wise gradient clipping is applied for numerical stability. QAM-E augments QAM with an additional residual edit policy $\pi_\omega(\Delta a \mid s, \tilde{a})$ that perturbs the QAM-generated action $\tilde{a}$ by at most σ_a in L_∞ distance (enforced by a tanh-squashed Gaussian), trained via entropy-regularized SAC with automatic entropy tuning. The difference from TRQAM is that they keep $\lambda = 1$ with a fixed β, whereas TRQAM adapts λ by projected dual descent.

C Experimental details

C.1 Domains and tasks

We evaluate on two benchmarks, OGBench [40] and Robomimic [32].

OGBench is originally designed for offline goal-conditioned RL, and we use its reward-based single-task variants. From OGBench, we use 10 domains spanning long-horizon navigation, multi-object manipulation, and combinatorial planning. Their abbreviations are `scene`, `puzzle-3x3` (`p33`), `puzzle-4x4` (`p44`), `cube-double` (`c2`), `cube-triple` (`c3`), `cube-quadruple` (`c4`), `humanoidmaze-medium` (`hm`), `humanoidmaze-large` (`hl`), `antmaze-large` (`al`), and `antmaze-giant` (`ag`).

Robomimic is a demonstration-based manipulation benchmark used as a stability stress-test. We use the `lift`, `can`, and `square` tasks.

The dataset size, episode length, and action dimension for each domain are reported in Table 3. For each method and task, we run 8 random seeds. Unless otherwise stated, tables report mean success rate $\pm$ standard deviation across seeds, and plots show the mean with shaded regions denoting standard deviation.

Dataset structure. Unlike teleoperation-style benchmarks where each demonstration directly solves the target task, OGBench's offline data is task-agnostic. The `navigate` datasets capture free maze exploration and `play` datasets capture unstructured object manipulation. The BC-pretrained policy thus serves as a behavioral prior rather than a task-specific solution.

Dataset sources. We use the official OGBench datasets [40] for all domains except where noted below. For `cube-triple-10M-*` and `puzzle-4x4-10M-*`, we use a 10M-size subset of the official 100M release. The 100M release is split into 100 files of 1M transitions each, and we take the first 10 files sorted by name, following [19]. For `antmaze-giant-10M-*`, OGBench does not release a dataset at the size used in our experiments, so we generate it ourselves using the official OGBench data-generation pipeline with default settings, and release it at https://huggingface.co/datasets/yonghoon96/antmaze-giant-navigate-10m-v0. For Robomimic, we use the Multi-Human (MH) datasets, each consisting of 300 trajectories collected by six operators of varying proficiency (two "worse", two "okay", and two "better"), yielding diverse mixed-quality demonstrations [32]. The behavior-cloning checkpoints used to initialize all methods on OGBench are available at https://huggingface.co/yonghoon96/trqam-bc-checkpoints.

C.2 Compute resources

Experiments ran on NVIDIA RTX 2080 Ti and A100-80GB GPUs, with a small share on RTX 3090, RTX A6000, and RTX 4090 cards. Table 2 reports the median wallclock time per run. The OGBench and Robomimic experiments reported in this paper used approximately 38,000 GPU-hours in total, excluding preliminary runs and the KL-budget sweeps used to select ε_{KL}.

Table 2: **Median wallclock time per run.** A100-80GB runs share one GPU among four runs.

	RTX 2080 Ti	A100-80GB
OGBench, 1M data	$\approx$ 7 h	$\approx$ 2–4 h
OGBench, 10M data	$\approx$ 19 h	$\approx$ 9 h
OGBench, 100M data	–	$\approx$ 7 h
Robomimic	$\approx$ 10–12 h	$\approx$ 11–12 h

C.3 Hyperparameters

Most methods share the common hyperparameters in Table 4, and method-specific hyperparameters are listed in Table 5.

OGBench tuning. For all baselines (FQL, DSRL, IFQL, CGQL-L, QAM, QAM-E), we adopt the per-domain values reported as optimal in QAM [24], which were selected through an extensive per-domain hyperparameter sweep covering 6–20 configurations per method across all ten domains, totaling approximately 32,000 GPU-hours. For TRQAM, we tune $\varepsilon_{\mathrm{KL}}$ on two tuning tasks per domain with 2 seeds per configuration (different from the evaluation seeds), sweeping over $\{0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$. Following Li and Levine [24], we use task 1 (the default task) and task 4 for locomotion domains, and task 2 (the default task) and task 4 for manipulation domains, as this combination better covers the characteristics of each domain than the default task alone. We select the configuration based on the combined offline-to-online learning curve and stability across the two tuning tasks. Selected values are reported in Table 5. For `antmaze-giant-10M-*`, we use a time-varying schedule with $\varepsilon_{\mathrm{KL}} = 0.5$ during offline training and 3.0 during online fine-tuning. The offline value is selected from the per-domain sweep, where it yielded the most stable offline training. However, retaining 0.5 online led to slow improvement in success rate, reflecting the larger exploration demands of giant-scale navigation. We therefore relax the bound at the online transition, selecting 3.0 from the same sweep range as a sweet spot between exploration and stability (see Appendix H.2 for a direct comparison of static vs. relaxed schedules). This schedule exploits TRQAM's capability to track time-varying $\varepsilon_{\mathrm{KL}}$ without retraining. All main-paper results are averaged over 8 evaluation seeds.

Robomimic tuning. Since Robomimic was not evaluated in the QAM paper [24], no per-domain hyperparameters are available, so we conducted an independent sweep for each method (DSRL, QAM, QAM-E, TRQAM) on this benchmark. As Robomimic serves as our stability testbed, we run the full sweep range with 8 seeds per configuration to obtain reliable estimates of variance across hyperparameters. For QAM-E, which has two hyperparameters, we sweep the full Cartesian product ($4 \times 2 = 8$ configurations). Robomimic consists of human teleoperation data, for which staying closer to behavior cloning is generally beneficial. We therefore adjust the QAM [24] sweep ranges toward configurations that more strongly anchor the policy to the behavior policy. The sweep ranges are listed in Table 6.

Table 3: **Domain metadata.**

Domain	**Data**	**Horizon**	**Act. dim.**
`cube-double-*`	1M	500	5
`cube-triple-10M-*`	10M	1000	5
`cube-quadruple-100M-*`	100M	1000	5
`antmaze-large-*`	1M	1000	8
`antmaze-giant-10M-*`	10M	1000	8
`humanoidmaze-medium-*`	1M	2000	21
`humanoidmaze-large-*`	1M	2000	21
`scene-*`	1M	750	5
`puzzle-3x3-*`	1M	500	5
`puzzle-4x4-10M-*`	10M	500	5
`lift`	31127	500	7
`can`	62756	500	7
`square`	80731	500	7

Table 4: **Common hyperparameters.**

Parameter	**Value**
Batch size	256
Discount factor (γ)	0.995 (default), 0.999 (`humanoidmaze`)
Optimizer	Adam
Learning rate	3×10^{-4}
Target network update rate	5×10^{-3}
Critic ensemble size (K)	10
Critic pessimism coefficient (ρ)	0.5 (default), 0 (`humanoidmaze`)
UTD ratio	1
Number of flow steps (T)	10
BC training steps	0.3×10^6
Offline RL steps	10^6
Online RL steps	0.5×10^6
Network width	512 (default), 1024 (10M/100M data)
Network depth	4 hidden layers
Gradient max-norm clipping	False (default), 1 (`QAM`, `QAM-E`, `TRQAM`)
Actor layer norm	False (default), True (10M/100M data)
Critic layer norm	True
Initial trust-region parameter (λ_0)	1
Dual step size (η_λ)	0.01
Range of λ	$[0.01, 100]$
KL EMA coefficient (ρ_D)	0.1

Table 5: **Domain-specific hyperparameters.**

Domain	`FQL` α	`DSRL` σ_z	`IFQL` κ	`CGQL-L` $(\vartheta, \varrho, \tau)$	`QAM` β	`QAM-E` (β, σ_a)	`TRQAM` $\varepsilon_{\mathrm{KL}}$
`scene-*`	300	0.4	0.9	$(10, 0.1, 0.1)$	1	$(1, 0)$	0.5
`puzzle-3x3-*`	300	1.0	0.95	$(10, 0.001, 0.1)$	3	$(1, 0.1)$	2.0
`puzzle-4x4-10M-*`	1	1.0	0.9	$(10, 0.001, 1)$	30	$(0.1, 0.9)$	4.0
`cube-double-*`	300	1.0	0.9	$(10, 0.001, 0.01)$	1	$(1, 0)$	0.5
`cube-triple-10M-*`	30	1.4	0.95	$(10, 0.001, 0.1)$	3	$(3, 0.1)$	0.5
`cube-quadruple-100M-*`	100	1.4	0.95	$(10, 0.1, 0.01)$	1	$(3, 0.1)$	1.0
`antmaze-large-*`	3	0.8	0.9	$(10, 0.001, 0.1)$	10	$(1, 0.1)$	1.0
`antmaze-giant-10M-*`	3	1.2	0.8	$(10, 0.001, 0.1)$	3	$(10, 0.1)$	$0.5 \rightarrow 3.0$
`humanoidmaze-medium-*`	30	0.6	0.7	$(10, 0.1, 0.1)$	3	$(3, 0.1)$	0.5
`humanoidmaze-large-*`	30	0.8	0.8	$(10, 0.1, 0.1)$	3	$(3, 0.1)$	0.5
`lift`	-	0.8	-	-	1	$(1, 0.03)$	0.1
`can`	-	0.6	-	-	0.3	$(1, 0.03)$	0.1
`square`	-	0.8	-	-	0.1	$(1, 0.03)$	0.5

Table 6: **Robomimic hyperparameter sweep ranges.**

Method	**Hyperparameter(s)**	**Sweep Range**
`DSRL`	σ_z	$\{0.03, 0.1, 0.2, 0.4, 0.6, 0.8\}$
`QAM`	β	$\{0.01, 0.03, 0.1, 0.3, 1, 3\}$
`QAM-E`	(β, σ_a)	$(\{0.03, 0.1, 0.3, 1\},\ \{0.03, 0.1\})$
`TRQAM`	$\varepsilon_{\mathrm{KL}}$	$\{0.01, 0.03, 0.1, 0.5, 1.0, 1.5\}$

D Proofs

This section provides full proofs for the four theoretical results stated in Section 3 of the main text, namely Theorem 1 (path-space KL identity), Proposition 1 (Terminal KL upper-bounded by path-space KL), Proposition 2 (tilted-policy KL is non-decreasing in β), and Lemma 1 (exponential amplification of critic errors).

We work throughout on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_\tau)_{\tau\in[0,1]}, \mathbb{P})$ equipped with a natural filtration and an initial state distribution $X_0 \sim p_0$. We adopt standard regularity assumptions for controlled diffusions, as in Nüsken and Richter [36]. Specifically, the coefficients b and σ are sufficiently smooth, b has at most linear growth, $\sigma\sigma^\top$ is uniformly positive definite, and the admissible control set $\mathcal{U}$ consists of progressively measurable controls with at most linear growth. Crucially, to ensure strong duality in our KL-budgeted improvement problem, we assume that the set of achievable path measures $\{\mathbb{P}^u \mid u \in \mathcal{U}\}$ is convex. Moreover, we assume that $u \in \mathcal{U}$ satisfies Novikov's condition,

$$\mathbb{E}_{\mathbf{X}\sim\mathbb{P}^{\text{base}}}\left[\exp\left(\frac{1}{2\lambda}\int_0^1 \|u(X_\tau,\tau)\|^2\,d\tau\right)\right] < \infty,$$

which justifies the application of Girsanov's theorem in the proof of Theorem 1.

D.1 Proof of Lemma 1 (exponential amplification of critic errors)

Lemma 1 (Exponential amplification of critic errors)**.** *Fix a state $s \in \mathcal{S}$. Let $Q : \mathcal{A} \to \mathbb{R}$ be the true critic and $\widetilde{Q} : \mathcal{A} \to \mathbb{R}$ a learned critic with $\|Q - \widetilde{Q}\|_\infty \le \varepsilon$. Assume $\pi_{\text{base}}(\cdot \mid s) > 0$ a.e. and define the policies obtained by exponentially tilting π_{base} with each critic,*

$$\pi_Q(a \mid s) = \frac{\pi_{\text{base}}(a \mid s)\, e^{\beta Q(a)}}{Z_Q}, \qquad \pi_{\widetilde{Q}}(a \mid s) = \frac{\pi_{\text{base}}(a \mid s)\, e^{\beta \widetilde{Q}(a)}}{Z_{\widetilde{Q}}},$$

with normalizers $Z_Q, Z_{\widetilde{Q}} \in (0, \infty)$. Then

$$\mathrm{TV}(\pi_Q, \pi_{\widetilde{Q}}) \le \frac{1}{2}\big(e^{2\beta\varepsilon} - 1\big), \qquad D_{\mathrm{KL}}(\pi_Q \,\|\, \pi_{\widetilde{Q}}) \le 2\beta\varepsilon.$$

Proof. For notational simplicity, we suppress s throughout the proof. From $\|Q - \widetilde{Q}\|_\infty \le \varepsilon$, multiplying $-\varepsilon \le Q(a) - \widetilde{Q}(a) \le \varepsilon$ by β and exponentiating yields, for all a,

$$e^{-\beta\varepsilon} \le e^{\beta(Q(a)-\widetilde{Q}(a))} \le e^{\beta\varepsilon}. \tag{8}$$

Multiplying Equation (8) by $\pi_{\text{base}}(a)\, e^{\beta\widetilde{Q}(a)} \ge 0$ and integrating gives $e^{-\beta\varepsilon} Z_{\widetilde{Q}} \le Z_Q \le e^{\beta\varepsilon} Z_{\widetilde{Q}}$. Combining this with Equation (8) in the likelihood ratio

$$\frac{\pi_Q(a)}{\pi_{\widetilde{Q}}(a)} = e^{\beta(Q(a)-\widetilde{Q}(a))} \cdot \frac{Z_{\widetilde{Q}}}{Z_Q}$$

shows that both factors lie in $[e^{-\beta\varepsilon}, e^{\beta\varepsilon}]$, so

$$e^{-2\beta\varepsilon} \;\le\; \frac{\pi_Q(a)}{\pi_{\widetilde{Q}}(a)} \;\le\; e^{2\beta\varepsilon} \quad \text{a.e.} \tag{9}$$

The TV bound follows from Equation (9), which gives $|\pi_Q(a) - \pi_{\widetilde{Q}}(a)| \le (e^{2\beta\varepsilon} - 1)\,\pi_{\widetilde{Q}}(a)$ a.e., and integrating with $\int \pi_{\widetilde{Q}} = 1$ gives $\mathrm{TV}(\pi_Q, \pi_{\widetilde{Q}}) = \frac{1}{2}\|\pi_Q - \pi_{\widetilde{Q}}\|_1 \le \frac{1}{2}(e^{2\beta\varepsilon} - 1)$. The KL bound also follows from Equation (9), which gives $\log(\pi_Q/\pi_{\widetilde{Q}}) \le 2\beta\varepsilon$ a.e., so

$$D_{\mathrm{KL}}(\pi_Q \,\|\, \pi_{\widetilde{Q}}) = \int \pi_Q(a) \log \frac{\pi_Q(a)}{\pi_{\widetilde{Q}}(a)}\, da \le 2\beta\varepsilon \int \pi_Q(a)\, da = 2\beta\varepsilon.$$

□

D.2 Proof of Theorem 1 (path-space KL identity)

Theorem 1 (Path-space KL identity with explicit λ dependence)**.** *Let $\mathbb{P}^{\text{base}}$ and $\mathbb{P}^u$ denote the path measures on $C([0,1];\mathbb{R}^d)$ induced by the base and controlled SDEs, respectively,*

$$dX_\tau^{\text{base}} = b(X_\tau^{\text{base}},\tau)\,d\tau + \sqrt{\lambda}\,\sigma(\tau)\,dB_\tau^{\text{base}},$$
$$dX_\tau^u = \big(b(X_\tau^u,\tau) + \sigma(\tau)\,u(X_\tau^u,\tau)\big)\,d\tau + \sqrt{\lambda}\,\sigma(\tau)\,dB_\tau^u,$$

with common initial distribution $X_0 \sim p_0$. Then

$$D_{\mathrm{KL}}\big(\mathbb{P}^u(\mathbf{X} \mid X_0) \,\|\, \mathbb{P}^{\text{base}}(\mathbf{X} \mid X_0)\big) = \mathbb{E}_{\mathbf{X}\sim\mathbb{P}^u}\left[\frac{1}{2\lambda}\int_0^1 \|u(X_\tau^u,\tau)\|^2\,d\tau\right].$$

Proof. Let $(\mathcal{F}_\tau)_{\tau\in[0,1]}$ denote the natural filtration on the canonical path space $C([0,1];\mathbb{R}^d)$. By Girsanov's theorem [39, Theorem 8.6.6], the Radon–Nikodym derivative of $\mathbb{P}^u$ with respect to $\mathbb{P}^{\text{base}}$ restricted to $\mathcal{F}_\tau$ is

$$Z_\tau := \left.\frac{d\mathbb{P}^u}{d\mathbb{P}^{\text{base}}}\right|_{\mathcal{F}_\tau} = \exp\left(\frac{1}{\sqrt{\lambda}}\int_0^\tau u(X_s^u,s)^\top\,dB_s^{\text{base}} - \frac{1}{2\lambda}\int_0^\tau \|u(X_s^u,s)\|^2\,ds\right), \tag{10}$$

where B^{base} is a standard Brownian motion under $\mathbb{P}^{\text{base}}$. By the definition of KL divergence on the full path space (terminal σ-algebra $\mathcal{F}_1$),

$$D_{\mathrm{KL}}\big(\mathbb{P}^u \,\|\, \mathbb{P}^{\text{base}}\big) = \mathbb{E}_{\mathbf{X}\sim\mathbb{P}^u}[\log Z_1].$$

Next, define

$$B_\tau^u := B_\tau^{\text{base}} - \frac{1}{\sqrt{\lambda}}\int_0^\tau u(X_s^u,s)\,ds.$$

By Girsanov's theorem, B^u is a standard Brownian motion under $\mathbb{P}^u$. In differential form, $dB_s^{\text{base}} = dB_s^u + \frac{1}{\sqrt{\lambda}}\,u(X_s^u,s)\,ds$. Substituting this into Equation (10) at $\tau = 1$ gives

$$\begin{aligned}\log Z_1 &= \frac{1}{\sqrt{\lambda}}\int_0^1 u(X_s^u,s)^\top\left(dB_s^u + \frac{1}{\sqrt{\lambda}}\,u(X_s^u,s)\,ds\right) - \frac{1}{2\lambda}\int_0^1 \|u(X_s^u,s)\|^2\,ds \\ &= \frac{1}{\sqrt{\lambda}}\int_0^1 u(X_s^u,s)^\top\,dB_s^u + \frac{1}{2\lambda}\int_0^1 \|u(X_s^u,s)\|^2\,ds.\end{aligned}$$

Taking expectation under $\mathbb{P}^u$ and using that the Itô integral $\int_0^1 u(X_s^u,s)^\top\,dB_s^u$ is a zero-mean martingale yields

$$D_{\mathrm{KL}}\big(\mathbb{P}^u \,\|\, \mathbb{P}^{\text{base}}\big) = \mathbb{E}_{\mathbf{X}\sim\mathbb{P}^u}\left[\frac{1}{2\lambda}\int_0^1 \|u(X_s^u,s)\|^2\,ds\right],$$

which is the claimed identity. □

Comparison to prior SOC parameterizations. The SOC parameterization used in prior work [9] takes the form

$$dX_\tau^u = \big(b(X_\tau^u,\tau) + \sigma(\tau)\,u(X_\tau^u,\tau)\big)\,d\tau + \sigma(\tau)\,dB_\tau^u,$$

where $\sigma(\tau)$ is the full diffusion and no λ appears. Girsanov's theorem then yields

$$D_{\mathrm{KL}}(\mathbb{P}^u\|\mathbb{P}^{\text{base}}) = \mathbb{E}_{\mathbf{X}\sim\mathbb{P}^u}\left[\frac{1}{2}\int_0^1 \|u(X_s^u,s)\|^2\,ds\right].$$

In this form, the path-space KL equals the SOC quadratic cost, and no parameter inside the SDE modulates the strength of this regularization. Adapting the trust-region strength would then require an external coefficient on the SOC objective. Our parameterization instead lets λ scale the control term in the drift, which makes λ the exact weight of the path-space KL in the SOC objective. This is what allows λ to serve as an adaptive dual variable for KL-budgeted improvement (Section 3).

D.3 Proof of Proposition 1 (terminal KL upper-bounded by path-space KL)

Proposition 1 (Terminal KL upper-bounded by path-space KL)**.** *Let $\mathbb{P}^{\mathrm{base}}$ and $\mathbb{P}^u$ denote the path measures on $C([0,1];\mathbb{R}^d)$ induced by the base and controlled SDEs, respectively,*

$$dX_\tau^{\mathrm{base}} = b(X_\tau^{\mathrm{base}},\tau)\,d\tau + \sqrt{\lambda}\,\sigma(\tau)\,dB_\tau^{\mathrm{base}},$$
$$dX_\tau^{u} = \big(b(X_\tau^{u},\tau) + \sigma(\tau)\,u(X_\tau^{u},\tau)\big)\,d\tau + \sqrt{\lambda}\,\sigma(\tau)\,dB_\tau^{u},$$

with common initial distribution $X_0 \sim p_0$. Let $\pi_\theta(\cdot \mid s)$ and $\pi_{\mathrm{base}}(\cdot \mid s)$ denote the corresponding terminal action distributions at $\tau = 1$, respectively. Then

$$D_{\mathrm{KL}}\big(\pi_\theta(\cdot \mid s) \,\|\, \pi_{\mathrm{base}}(\cdot \mid s)\big) \;\le\; D_{\mathrm{KL}}\big(\mathbb{P}^u(\mathbf{X} \mid X_0) \,\|\, \mathbb{P}^{\mathrm{base}}(\mathbf{X} \mid X_0)\big),$$

Proof. Let $\Pi : C([0,1];\mathbb{R}^d) \to \mathbb{R}^d$ denote the deterministic terminal projection $\Pi(\mathbf{X}) = X_1$. By construction, the terminal action distributions are pushforwards of the path measures, namely $\pi_\theta(\cdot \mid s) = \Pi_{\#}\,\mathbb{P}^u$ and $\pi_{\mathrm{base}}(\cdot \mid s) = \Pi_{\#}\,\mathbb{P}^{\mathrm{base}}$. Applying the data-processing inequality for KL divergence under maps [46, Corollary. 2.18] to Π yields

$$D_{\mathrm{KL}}\big(\Pi_{\#}\,\mathbb{P}^u \,\|\, \Pi_{\#}\,\mathbb{P}^{\mathrm{base}}\big) \;\le\; D_{\mathrm{KL}}\big(\mathbb{P}^u \,\|\, \mathbb{P}^{\mathrm{base}}\big)\,,$$

which proves the claim. □

D.4 Proof of Proposition 2 (tilted-policy KL is non-decreasing in β)

Proposition 2 (Tilted-policy KL is non-decreasing in β)**.** *Fix a state $s \in \mathcal{S}$ and a bounded critic $Q : \mathcal{A} \to \mathbb{R}$. For $\beta > 0$, define the exponentially tilted policy and its log-partition function*

$$\pi_\beta(a \mid s) = \frac{\pi_{\mathrm{base}}(a \mid s)\, e^{\beta Q(a)}}{Z(\beta)}, \qquad \psi(\beta) = \log Z(\beta),$$

where $Z(\beta) = \mathbb{E}_{a\sim\pi_{\mathrm{base}}}[e^{\beta Q(a)}]$. Then the following identities hold

$$D_{\mathrm{KL}}(\pi_\beta \,\|\, \pi_{\mathrm{base}}) = \beta\psi'(\beta) - \psi(\beta), \qquad \frac{d}{d\beta} D_{\mathrm{KL}}(\pi_\beta \,\|\, \pi_{\mathrm{base}}) = \beta \mathrm{Var}_{\pi_\beta}[Q] \ge 0.$$

In particular, $D_{\mathrm{KL}}(\pi_\beta \,\|\, \pi_{\mathrm{base}})$ is non-decreasing in β.

Proof. Since Q is bounded, $Z(\beta)$ is finite and positive for every β, and differentiation under the expectation is justified. Differentiating ψ gives the first two moments of Q under π_β,

$$\psi'(\beta) = \frac{\mathbb{E}_{\pi_{\mathrm{base}}}[Q\, e^{\beta Q}]}{Z(\beta)} = \mathbb{E}_{\pi_\beta}[Q], \qquad \psi''(\beta) = \frac{\mathbb{E}_{\pi_{\mathrm{base}}}[Q^2 e^{\beta Q}]}{Z(\beta)} - \psi'(\beta)^2 = \mathrm{Var}_{\pi_\beta}[Q].$$

The log-density ratio is $\log\big(\pi_\beta(a \mid s)/\pi_{\mathrm{base}}(a \mid s)\big) = \beta Q(a) - \psi(\beta)$. Taking its expectation under π_β yields

$$D_{\mathrm{KL}}(\pi_\beta \,\|\, \pi_{\mathrm{base}}) = \beta\, \mathbb{E}_{\pi_\beta}[Q] - \psi(\beta) = \beta\psi'(\beta) - \psi(\beta).$$

Differentiating in β gives

$$\frac{d}{d\beta} D_{\mathrm{KL}}(\pi_\beta \,\|\, \pi_{\mathrm{base}}) = \psi'(\beta) + \beta\psi''(\beta) - \psi'(\beta) = \beta \mathrm{Var}_{\pi_\beta}[Q] \ge 0.$$

Hence the divergence is non-decreasing in β. □

E Path-space KL surrogate under OT memoryless discretization

This section derives the closed-form path-space KL estimator $\widehat{D}_n$ used in Algorithm 3. Recall from Theorem 1 that the path-space KL between $\mathbb{P}^u$ and $\mathbb{P}^{\mathrm{base}}$ admits a closed-form expression in terms of the control u, and from Proposition 1 that this path-space KL upper-bounds the terminal-policy KL we ultimately wish to constrain. We thus seek a tractable estimator of the path-space KL under the discretized OT memoryless sampler.

Gaussian KL with shared covariance. For two Gaussians $\mathcal{N}(\mu_1, \Sigma)$ and $\mathcal{N}(\mu_0, \Sigma)$ with shared covariance $\Sigma \succ 0$, specializing the multivariate Gaussian KL formula [46, Equation. 2.8] makes the log-determinant and trace terms vanish. In other words,

$$D_{\mathrm{KL}}(\mathcal{N}(\mu_1, \Sigma) \,\|\, \mathcal{N}(\mu_0, \Sigma)) = \tfrac{1}{2}\,(\mu_1 - \mu_0)^\top \Sigma^{-1} (\mu_1 - \mu_0). \tag{11}$$

OT memoryless Euler step KL. Under the OT memoryless Euler scheme with step size h and schedule $\nu(\tau) = \sqrt{2(1-\tau)/\tau}$, the fine-tuned and base transitions at time τ_k given $X_{\tau_k} = x$ are both Gaussian with covariance $\Sigma_k = h\,\nu(\tau_k)^2\, I$ and means

$$\mu_\theta(x, \tau_k) = x + h\Big(2v_\theta^{\mathrm{ft}}(x, \tau_k) - \tfrac{1}{\tau_k}\, x\Big), \qquad \mu_{\mathrm{base}}(x, \tau_k) = x + h\Big(2v^{\mathrm{base}}(x, \tau_k) - \tfrac{1}{\tau_k}\, x\Big).$$

The means differ only through the velocity fields, since $\mu_\theta(x, \tau_k) - \mu_{\mathrm{base}}(x, \tau_k) = 2h\,(v_\theta^{\mathrm{ft}}(x, \tau_k) - v^{\mathrm{base}}(x, \tau_k))$. Substituting into Equation (11) with $\Sigma_k^{-1} = \frac{1}{h\,\nu(\tau_k)^2}\, I$ gives the per-step KL

$$D_{\mathrm{KL}}(p_\theta(\cdot \mid x) \,\|\, p_{\mathrm{base}}(\cdot \mid x)) = \frac{2h}{\nu(\tau_k)^2}\, \|v_\theta^{\mathrm{ft}}(x, \tau_k) - v^{\mathrm{base}}(x, \tau_k)\|^2. \tag{12}$$

Chain rule and Monte Carlo estimator. Let $\mathbb{P}^u$ and $\mathbb{P}^{\mathrm{base}}$ now denote the discrete-time path measures of the Markov chains $\boldsymbol{X} = (X_\tau)_{\tau \in \{0, h, \ldots, 1\}}$ with shared initial distribution p_0 and the Gaussian transition kernels above. Under mild regularity conditions, the Markov KL chain rule gives

$$D_{\mathrm{KL}}(\mathbb{P}^u \,\|\, \mathbb{P}^{\mathrm{base}}) = \sum_{\tau \in \{0, h, \ldots, 1-h\}} \mathbb{E}_{X_\tau \sim \mathbb{P}^u}\left[D_{\mathrm{KL}}(p_\theta(\cdot \mid X_\tau) \,\|\, p_{\mathrm{base}}(\cdot \mid X_\tau))\right].$$

Substituting Equation (12) and estimating the expectation by a Monte Carlo average over the trajectories in batch $\mathcal{B}$ yields

$$\widehat{D} = \frac{1}{|\mathcal{B}|} \sum_{\boldsymbol{X} \in \mathcal{B}} \sum_{\tau \in \{0, h, \ldots, 1-h\}} \frac{2h}{\nu(\tau)^2}\, \|v_\theta^{\mathrm{ft}}(X_\tau, \tau) - v^{\mathrm{base}}(X_\tau, \tau)\|^2, \tag{13}$$

which is the estimator used in Algorithm 3 (optionally smoothed by EMA to reduce variance).

F Primal–dual derivation of KL-budgeted improvement

This section derives the projected dual update on λ used in Algorithm 3.

Primal problem and Slater's condition. We consider the path-space KL-budgeted improvement problem

$$\begin{aligned} \max_{u \in \mathcal{U}} \quad & \mathbb{E}_{X \sim \mathbb{P}^u}[Q^\pi(X_1)] \\ \text{s.t.} \quad & D_{\mathrm{KL}}(\mathbb{P}^u \,\|\, \mathbb{P}^{\mathrm{base}}) \le \varepsilon_{\mathrm{KL}}. \end{aligned}$$

Viewed in the path measure space, the objective is linear in $\mathbb{P}^u$ and $D_{\mathrm{KL}}(\cdot \,\|\, \mathbb{P}^{\mathrm{base}})$ is convex. The trivial control $u \equiv 0$ recovers $\mathbb{P}^u = \mathbb{P}^{\mathrm{base}}$ and is strictly feasible whenever $\varepsilon_{\mathrm{KL}} > 0$, so Slater's condition holds and strong duality applies.

Lagrangian and dual function. Let $\lambda \ge 0$ be the dual variable for the KL constraint. The Lagrangian is

$$\mathcal{L}(u, \lambda) = \mathbb{E}_{X \sim \mathbb{P}^u}[Q^\pi(X_1)] + \lambda\big(\varepsilon_{\mathrm{KL}} - D_{\mathrm{KL}}(\mathbb{P}^u \,\|\, \mathbb{P}^{\mathrm{base}})\big),$$

and the dual function is $\Gamma(\lambda) := \sup_u \mathcal{L}(u, \lambda)$. By strong duality, minimizing Γ over $\lambda \ge 0$ solves the primal. A standard subgradient of Γ at λ is

$$s(\lambda) = \varepsilon_{\mathrm{KL}} - D_{\mathrm{KL}}(\mathbb{P}^{u_\lambda} \,\|\, \mathbb{P}^{\mathrm{base}}),$$

where $u_\lambda := \arg\sup_u \mathcal{L}(u, \lambda)$ denotes the inner maximizer at the current dual variable. The subgradient inequality follows immediately. For any $\lambda' \ge 0$,

$$\Gamma(\lambda') \ge \mathcal{L}(u_\lambda, \lambda') = \Gamma(\lambda) + (\lambda' - \lambda)\big(\varepsilon_{\mathrm{KL}} - D_{\mathrm{KL}}(\mathbb{P}^{u_\lambda} \,\|\, \mathbb{P}^{\mathrm{base}})\big).$$

Projected dual descent. Projected subgradient descent on $\min_{\lambda \geq 0} \Gamma(\lambda)$ would require evaluating the KL at the inner maximizer u_λ, which is not available in closed form. In Algorithm 3, we instead evaluate the path-space KL at the control u_n at iteration n via the Monte Carlo surrogate $\widehat{D}_n$ from Equation (13). The dual update

$$\lambda_{n+1} \leftarrow \max\{0,\ \lambda_n + \eta_\lambda(\overline{D}_n - \varepsilon_{\mathrm{KL}})\}$$

preserves the key sign property. When the realized KL exceeds the budget, λ rises and the controlled SDE becomes more conservative. When it falls below the budget, λ decreases and the policy can move further from π_{base} within the same budget.

G Detailed comparison of internal and external KL regularization

Section 3.4 contrasts two ways of pairing the dual update in Equation (7) with a KL constraint, namely appending $\overline{D}_n$ as an auxiliary loss term (*external*), or letting λ scale the control term in the drift inside the SOC sampling process (*internal*, TRQAM). The two formulations share the same dual update rule and the same KL estimator, yet they differ in how the constraint is enforced. This appendix expands on that contrast and explains why the difference is structural rather than cosmetic.

The key asymmetry is in what λ represents. In the external formulation, λ is a scalar weight on an auxiliary KL regularization term in the loss. It balances this regularization against the critic signal, but does not appear in the controlled SDE itself. The path-space KL between $\mathbb{P}^u$ and $\mathbb{P}^{\mathrm{base}}$ is therefore independent of λ, and is determined entirely by the learned control u. Adapting λ only changes the relative weight between the adjoint-matching loss $\mathcal{L}_{\text{Adj-Match}}$ and the KL regularization $\overline{D}_n$ during gradient descent. Whether the realized KL ends up close to $\varepsilon_{\mathrm{KL}}$ depends on the optimizer balancing these competing signals. Under strong critic guidance, the KL term can be effectively overridden, and the realized KL drifts above the target KL bound $\varepsilon_{\mathrm{KL}}$ (Figures 15 and 16).

In the internal formulation, by contrast, λ is a parameter of the controlled SDE itself. Theorem 1 establishes that the path-space KL is exactly $\mathbb{E}[\frac{1}{2\lambda}\int \|u\|^2]$, so λ appears as the inverse coefficient of the KL itself. Increasing λ shrinks the path-space KL directly through the control term in the drift, rather than indirectly through the critic gradient. The dual update therefore reshapes the trajectory distribution structurally rather than competing with critic guidance. Table 7 summarizes these distinctions.

Table 7: **Internal vs. external KL regularization.** Although both approaches can use the same dual update rule on λ, the role of λ differs structurally. Internalizing λ inside the SDE makes λ the exact weight of the path-space KL in the SOC objective (Theorem 1), turning the target KL bound $\varepsilon_{\mathrm{KL}}$ into a structural constraint rather than a soft penalty.

	External KL (auxiliary regularization)	**Internal KL** (TRQAM)
Controlled SDE	$dX_\tau^u = (b + \sigma u)\, d\tau + \sigma(\tau)\, dB_\tau$	$dX_\tau^u = (b + \sigma u)\, d\tau + \sqrt{\lambda}\,\sigma(\tau)\, dB_\tau$
Path-space KL	$\mathbb{E}\left[\frac{1}{2}\int_0^1 \|u\|^2\, d\tau\right]$ (λ-independent)	$\mathbb{E}\left[\frac{1}{2\lambda}\int_0^1 \|u\|^2\, d\tau\right]$ (Theorem 1)
Role of λ	Regularization weight on auxiliary loss	Intrinsic SDE parameter
Training loss	$\mathcal{L}_{\text{Adj-Match}}(\theta) + \lambda\, \overline{D}_n(\theta)$	$\mathcal{L}_{\text{Adj-Match}}(\theta)$
Effect of increasing λ	Competes with critic guidance at loss level	Reshapes the SDE at sampling level
Enforcement mechanism	Soft (gradient competition)	Structural (via the sampling process)
Realized KL vs. target bound	Exceed the target bound (Figures 15 and 16)	Tracks target bound tightly

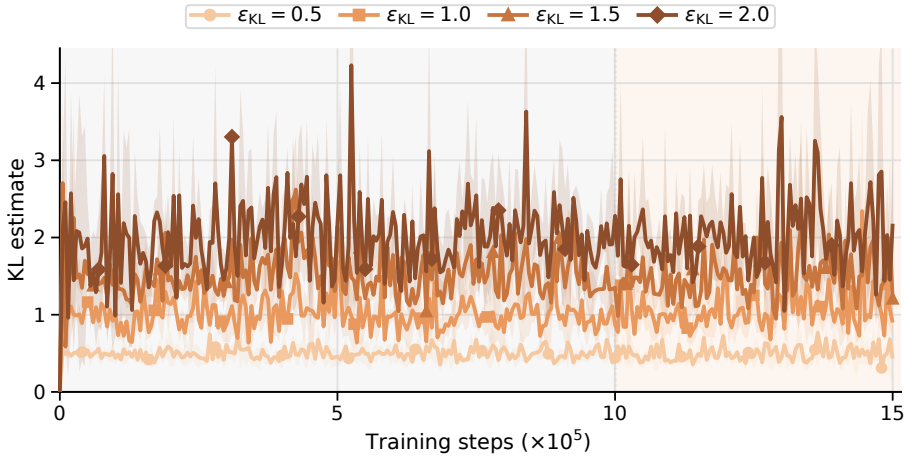


Figure 8: **TRQAM tracks the prescribed KL budget across offline and online training.** We vary the target KL bound $\varepsilon_{\mathrm{KL}} \in \{0.5, 1.0, 1.5, 2.0\}$ and plot the realized path-space KL throughout training. Larger budgets produce correspondingly larger policy deviation, and the monotonic ordering is preserved across the offline-to-online transition.

H Additional experiments

H.1 KL-budget tracking

TRQAM faithfully enforces the prescribed KL budget across a range of target values. Figure 8 plots the realized path-space KL under four budgets $\varepsilon_{\mathrm{KL}} \in \{0.5, 1.0, 1.5, 2.0\}$ throughout offline and online training. The realized KL tracks each prescribed target with a clear monotonic ordering, and this ordering is preserved across the offline-to-online transition, indicating that the dual update remains effective under distribution shift. The KL constraint in TRQAM is therefore both controllable and stable across training regimes.

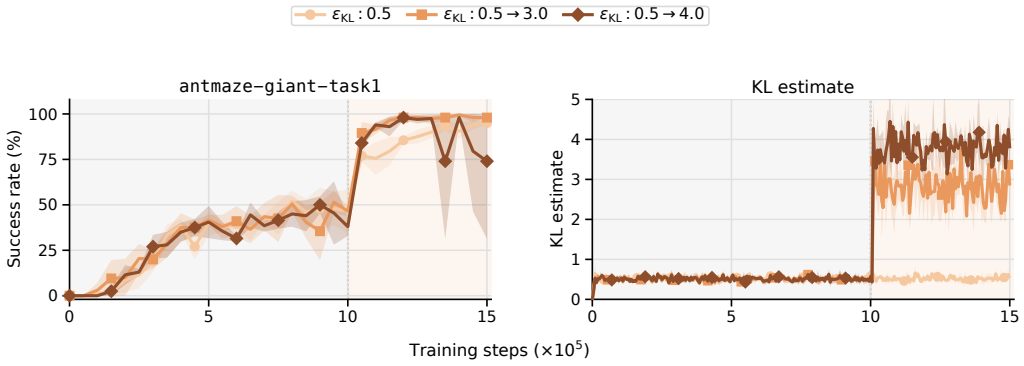


Figure 9: **TRQAM tracks time-varying $\varepsilon_{\mathrm{KL}}$ schedules across the offline-to-online transition.** We illustrate this on `antmaze-giant`, the domain in our benchmark with the largest state space and accordingly the highest exploration demands during online fine-tuning. We keep $\varepsilon_{\mathrm{KL}} = 0.5$ during offline training and optionally switch to a larger bound at the online transition (vertical dashed line). In the right panel, the realized KL adapts to each new target almost immediately, showing that the dual update on λ handles schedule changes without instability. In the left panel, the static schedule (0.5) improves more slowly online than the moderate schedule ($0.5 \to 3.0$), which reaches $\sim$98% success rapidly and outperforms the more aggressive ($0.5 \to 4.0$) alternative.

Tracking accuracy. To quantify how closely TRQAM tracks the budget, we record the realized path-space KL throughout training on one representative task per OGBench domain (`task1` for `antmaze` and `humanoidmaze`, `task2` for `scene` and `cube`, and `task4` for `puzzle`), with a common budget $\varepsilon_{\mathrm{KL}} = 0.5$ and 2 seeds. We measure the relative deviation $|\overline{D}_n - \varepsilon_{\mathrm{KL}}|/\varepsilon_{\mathrm{KL}}$ at every training step. Table 8 shows that the realized KL stays centered at the budget on all domains, with a mean relative deviation of 5.0% and a 95th-percentile deviation of 12.7%.

Table 8: **KL-budget tracking accuracy** at $\varepsilon_{\mathrm{KL}} = 0.5$ (one representative task per domain, 2 seeds).

	`al`	`ag`	`hm`	`hl`	`scene`	`p33`	`p44`	`c2`	`c3`	`c4`	Overall
Path-space KL (mean)	0.501	0.501	0.499	0.500	0.498	0.500	0.501	0.498	0.499	0.501	–
Relative deviation (mean)	5.2%	6.0%	4.3%	4.9%	4.3%	4.6%	4.3%	3.7%	6.0%	6.4%	5.0%
Relative deviation (95th pct.)	11.8%	15.0%	10.3%	12.2%	10.5%	11.1%	10.6%	8.8%	17.0%	15.8%	12.7%

H.2 Time-varying KL budget

The KL bound $\varepsilon_{\mathrm{KL}}$ is not necessarily a fixed quantity throughout training. In offline-to-online RL, the appropriate trust region may differ between phases, since online fine-tuning often requires more exploration than the offline phase permits. TRQAM accommodates this naturally because λ is internalized in the SOC sampling process rather than entering as a soft loss penalty. As a result, the dual update on λ remains stable under time-varying $\varepsilon_{\mathrm{KL}}$, allowing the bound to be re-set at any point without retraining or instability.

We illustrate this on `antmaze-giant`, the domain in our benchmark with the largest state space and accordingly the highest exploration demands during online fine-tuning. Figure 9 sweeps three schedules on `antmaze-giant-task1`, the tuning task for our per-domain $\varepsilon_{\mathrm{KL}}$ choice. These are the static schedule (0.5, matching the offline value) and two relaxations at the online transition ($0.5 \to 3.0$, $0.5 \to 4.0$). Two observations follow.

First, the realized KL adapts to each new target almost immediately after the switch, confirming that TRQAM tracks time-varying bounds without loss of stability.

Second, the static schedule (0.5) improves substantially more slowly online than the moderate schedule ($0.5 \to 3.0$). On domains with large state spaces such as `antmaze-giant`, online fine-tuning relies on exploring beyond the offline support, and a tight KL budget restricts this exploration and slows adaptation to the new state distribution. The moderate schedule reaches $\sim$98% success rapidly, while the more aggressive $0.5 \to 4.0$ introduces substantial variance, identifying 3.0 as a sweet spot.

H.3 Sensitivity analysis

We extend the four-task sweep of Section 4.3 to all ten OGBench domains, sweeping $\varepsilon_{\mathrm{KL}} \in \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$ on one representative task per domain with all other hyperparameters fixed. Results across 8 seeds appear in Figure 10, showing two patterns. Tight budgets are best on `humanoidmaze-medium`, `humanoidmaze-large`, `cube-double`, `cube-triple`, and `cube-quadruple`. Larger budgets monotonically improve performance on `puzzle-4x4`, in keeping with its notably larger state space.

Smaller $\varepsilon_{\mathrm{KL}}$ also produces slower online adaptation across the sweep, consistent with the dual update tightening the trust region. We observe the same effect on `antmaze-giant` (Appendix H.2), where the static 0.5 schedule slows online adaptation in a domain whose large state space demands broader exploration.

H.4 A single global KL budget

The main results use the per-domain budgets of Table 5. To measure how much this tuning matters, we rerun TRQAM with a single global budget $\varepsilon_{\mathrm{KL}} = 0.5$ on every OGBench task whose tuned budget differs from 0.5, with 8 seeds. Table 9 reports the aggregate success rate over all 50 tasks at 1M steps. The global budget reaches 63%, close to the 68% of per-domain tuning and well above the strongest baseline at 46%. Excluding `puzzle-4x4`, where the score drops from 99 to 62, the global

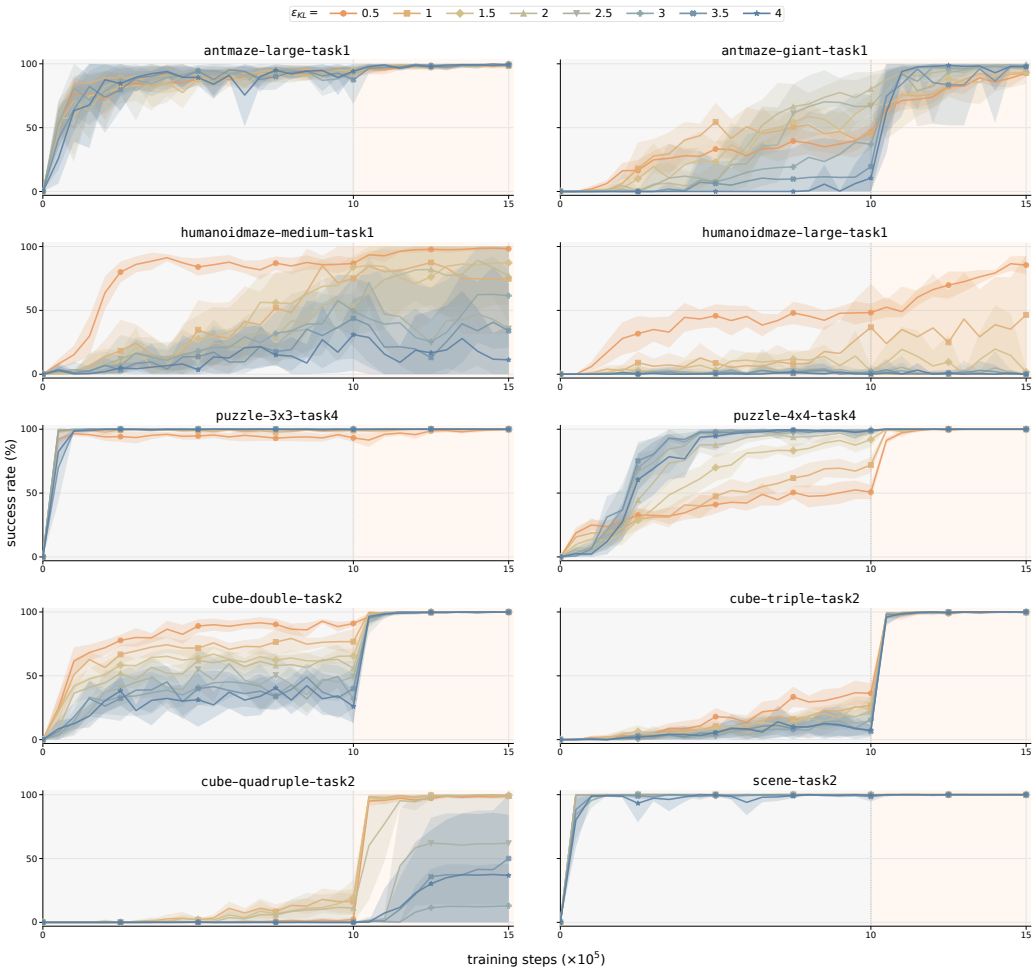


Figure 10: **Hyperparameter $\varepsilon_{\mathrm{KL}}$ sweep on OGBench [40] (8 seeds).** We sweep $\varepsilon_{\mathrm{KL}} \in \{0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$ on one representative task per domain, with all other hyperparameters fixed to the main-comparison setting. Each panel reports success rate over training steps, with the offline-to-online transition at 10^6 steps. Shaded regions denote ± 1 standard deviation across seeds. Evaluated tasks are listed in the panel titles. Two patterns emerge. Tight budgets are best on `humanoidmaze-medium`, `humanoidmaze-large`, `cube-double`, `cube-triple`, and `cube-quadruple`, while larger budgets monotonically improve performance on `puzzle-4x4`, consistent with its notably larger state space. Smaller $\varepsilon_{\mathrm{KL}}$ also produces slower online adaptation across the sweep, consistent with the dual update tightening the trust region.

budget stays within about one point of the tuned result. This exception matches Appendix H.3, where `puzzle-4x4` benefits monotonically from larger budgets because of its larger state space.

Table 9: **Per-domain versus a single global KL budget** (aggregate success rate over all 50 OGBench tasks, 8 seeds).

Budget	`al`	`ag`	`hm`	`hl`	`scene`	`p33`	`p44`	`c2`	`c3`	`c4`	Mean
Per-domain	89 ± 4	41 ± 4	84 ± 3	36 ± 4	79 ± 1	100 ± 0	99 ± 1	81 ± 3	50 ± 5	19 ± 5	68
Global $\varepsilon_{\mathrm{KL}} = 0.5$	89 ± 4	41 ± 4	84 ± 3	36 ± 4	79 ± 1	97 ± 3	62 ± 8	81 ± 3	50 ± 5	14 ± 3	63

H.5 Gap between the path-space KL and the terminal KL

Proposition 1 bounds the terminal KL by the path-space KL. By the chain rule of KL divergence, the gap is exactly [54]

$$D_{\mathrm{KL}}(\mathbb{P}^u \,\|\, \mathbb{P}^{\mathrm{base}}) = D_{\mathrm{KL}}(\pi_\theta \,\|\, \pi_{\mathrm{base}}) + \mathbb{E}_{X_1 \sim \pi_\theta}\big[D_{\mathrm{KL}}\big(\mathbb{P}^u(\cdot \mid X_1) \,\|\, \mathbb{P}^{\mathrm{base}}(\cdot \mid X_1)\big)\big],$$

which measures how differently the two processes reach the same action. To check how loose the bound is in practice, we measure both quantities on a Gaussian-mixture target in dimensions $d \in \{2, 8, 16\}$ under eight budgets. We estimate the terminal KL by the change of variables with a full Jacobian trace over 3000 samples [27]. As Table 10 shows, the terminal KL is never negligible. In all 24 configurations, it stays at least about 40% of the path-space KL, so the bound in Proposition 1 remains informative in practice.

Table 10: **Realized path-space KL / terminal KL** on a Gaussian-mixture target.

$d \backslash \varepsilon_{\mathrm{KL}}$	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
2	0.51/0.48	0.99/0.94	1.47/0.84	2.03/1.71	2.44/2.40	2.45/1.63	2.41/1.83	3.52/1.76
8	0.80/0.35	1.02/0.58	1.47/1.03	1.74/1.73	2.40/2.35	2.85/2.17	3.31/2.89	3.97/2.99
16	1.45/0.92	1.38/0.84	1.59/0.70	1.99/0.90	2.44/1.76	2.92/1.94	3.36/1.69	4.04/3.41

H.6 Controlling the deviation with λ versus β

Both β in QAM and λ in TRQAM move the policy away from π_{base} (Section 3.2). They differ in predictability. To compare them, we measure the realized path-space KL and the terminal KL on three 2D testbeds from Doo et al. [11] (8-gaussians, swiss-roll, and spiral), estimating the terminal KL by the change of variables over 9000 samples [27]. Under QAM (Table 11, top), the KL for a given β varies substantially across datasets and responds unevenly. It stays nearly flat from $\beta = 0.1$ to 1 and then rises sharply between $\beta = 3$ and 30. Setting β therefore gives no indication of how far the policy will move. Under TRQAM (bottom), the realized path-space KL instead follows the specified budget at a similar level on all three datasets. Setting $\varepsilon_{\mathrm{KL}}$ thus predicts how far the policy will move.

Table 11: **Realized path-space KL / terminal KL** under QAM with inverse temperature β (top) and TRQAM with budget $\varepsilon_{\mathrm{KL}}$ (bottom).

QAM	$\beta = 0.1$	0.3	1	3	10	30		
8-gaussians	0.16/0.02	0.17/0.02	0.25/0.10	1.15/0.99	3.38/3.33	4.59/3.48		
swiss-roll	0.15/0.04	0.15/0.03	0.20/0.04	0.76/0.48	2.95/1.46	7.92/4.04		
spiral	0.58/0.24	0.58/0.23	0.59/0.25	0.77/0.38	1.97/1.34	4.27/2.11		
TRQAM	$\varepsilon_{\mathrm{KL}} = 0.5$	1.0	1.5	2.0	2.5	3.0	3.5	4.0
8-gaussians	0.47/0.45	0.97/0.90	1.50/0.86	1.95/1.26	2.38/2.25	2.89/0.87	2.59/1.64	3.56/3.03
swiss-roll	0.50/0.27	1.01/0.46	1.52/1.02	2.06/1.63	2.64/2.02	2.39/0.77	2.72/0.63	3.16/1.36
spiral	0.74/0.34	1.07/0.49	1.49/1.45	1.99/1.03	2.38/1.64	2.56/1.26	3.15/2.18	3.92/2.41

H.7 Robustness to the dual-update hyperparameters

TRQAM uses a single setting of the dual step size η_λ and the EMA coefficient ρ_D across all tasks (Table 4). To test how much this choice matters, we vary each of them on one representative task per OGBench domain with 8 seeds. Table 12 reports the results. ρ_D has little effect, giving the same mean of 70.2 under heavy smoothing ($\rho_D = 0.01$) and no smoothing ($\rho_D = 1$). For η_λ, performance stays close to the default at 0.001 and drops only at 0.1. Because λ enters the SOC formulation directly, a large dual step makes λ react greedily to each KL estimate, so a small step size is the natural choice.

Table 12: **Sensitivity to the dual-update hyperparameters** (success rate on one representative task per domain, 8 seeds).

Setting	al	ag	hm	hl	scene	p33	p44	c2	c3	c4	Mean
Default ($\eta_\lambda = 0.01$, $\rho_D = 0.1$)	95 ± 4	48 ± 5	87 ± 5	45 ± 8	100 ± 0	100 ± 1	100 ± 1	92 ± 3	42 ± 9	18 ± 11	72.7
$\eta_\lambda = 0.001$	94 ± 4	37 ± 10	87 ± 5	50 ± 12	100 ± 0	100 ± 0	100 ± 1	88 ± 5	28 ± 8	14 ± 9	69.8
$\eta_\lambda = 0.1$	84 ± 10	4 ± 4	86 ± 5	41 ± 11	100 ± 0	100 ± 1	91 ± 7	87 ± 5	26 ± 12	1 ± 1	62.0
$\rho_D = 0.01$	94 ± 5	43 ± 6	86 ± 4	47 ± 8	100 ± 0	100 ± 0	99 ± 2	90 ± 8	29 ± 8	14 ± 8	70.2
$\rho_D = 1$	95 ± 3	45 ± 11	87 ± 8	42 ± 6	100 ± 0	100 ± 1	99 ± 2	84 ± 4	32 ± 8	18 ± 5	70.2

H.8 Number of critics

All methods use an ensemble of 10 critics (Table 4). Fewer critics lead to larger critic errors, which Lemma 1 predicts should hurt QAM more than TRQAM. To test this, we compare 2 and 10 critics on one representative task per OGBench domain with 8 seeds. Table 13 shows that reducing the ensemble lowers TRQAM by only 4 points ($73 \to 69$) but QAM by 7 points ($37 \to 30$), so the gap between them widens from 36 to 39 points. Even with only 2 critics, TRQAM stays more than 30 points above QAM with 10 critics. This is consistent with our claim that the trust region limits the effect of critic errors.

Table 13: **Effect of the number of critics** (success rate on one representative task per domain, 8 seeds).

Method	al	ag	hm	hl	scene	p33	p44	c2	c3	c4	Mean
QAM (10 critics)	96 ± 3	37 ± 9	33 ± 34	6 ± 10	100 ± 0	0 ± 0	0 ± 1	81 ± 6	15 ± 11	2 ± 4	37
QAM (2 critics)	74 ± 37	41 ± 10	13 ± 24	5 ± 10	100 ± 1	0 ± 1	0 ± 0	59 ± 12	5 ± 5	1 ± 2	30
TRQAM (10 critics)	95 ± 4	48 ± 5	87 ± 5	45 ± 8	100 ± 0	100 ± 1	100 ± 1	92 ± 3	42 ± 9	18 ± 11	73
TRQAM (2 critics)	90 ± 4	52 ± 11	89 ± 4	44 ± 10	100 ± 1	100 ± 0	100 ± 1	78 ± 7	24 ± 11	15 ± 11	69

H.9 Computational cost

Estimating the path-space KL reuses the velocity difference $v_\theta^{\mathrm{ft}} - v^{\mathrm{base}}$ already computed for the Adjoint Matching loss, so the trust region adds $O(1)$ time and memory per step over QAM. The main cost shared with QAM is the vector-Jacobian product (VJP) in the lean adjoint ODE. Tables 14 and 15 report its share of per-step training time. The share grows with the actor size, rising from 9.2% to 21.1% over a $240\times$ increase in parameters (Table 14). It does not grow without bound, but at about a fifth of each training step for larger actors, the VJP cost becomes hard to ignore as the policy grows (Appendix K). The action dimension has little effect (Table 15).

Table 14: **VJP share of per-step training time versus actor size** (action dimension 14).

Actor size	Time	FLOPs
0.84M	9.2%	13.1%
12.78M	19.4%	22.8%
202.12M	21.1%	24.0%

Table 15: **VJP share of per-step training time versus action dimension** for a small and a large actor.

Action dim.	0.84M actor	50.7M actor
5	10.2%	19.3%
21	9.8%	18.8%
128	10.1%	22.6%
1024	10.9%	19.8%

H.10 Stability stress test on Robomimic

We complement our OGBench results with Robomimic [32], a standard manipulation benchmark on which we observe fixed-temperature Adjoint Matching to be unstable.

Fixed-temperature methods collapse across the sweep. Sweeping QAM over six values of $\beta \in \{0.01, 0.03, 0.1, 0.3, 1.0, 3.0\}$ and QAM-E over eight (β, σ_a) configurations, the collapse pattern persists across most settings on `lift` and `can` (Figures 11 and 12), where the adjoint-matching loss explodes during offline training and success rate collapses to near zero. On `square`, the explosion is milder, but the adjoint loss of the fixed-temperature variants still grows steadily during offline training, while TRQAM keeps it bounded (Figure 13). This is consistent with the structural fragility predicted by fixed-β critic-error amplification (Lemma 1).

External KL is insufficient, while TRQAM enforces the budget. External KL regularization partially mitigates the collapse, but the realized path-space KL substantially exceeds the prescribed bound across every budget, leaving the policy vulnerable to critic-error amplification. TRQAM, by contrast, tracks each budget tightly throughout offline training and across the offline-to-online transition, and remains stable across the full sweep $\varepsilon_{\mathrm{KL}} \in \{0.01, 0.03, 0.1, 0.5, 1, 1.5\}$ (Figures 15 to 17).

Best configuration of each method. Figure 14 compares the best configuration of each method on all three tasks.

I Full OGBench results

Table 16 reports the offline success rate of every method on all 50 OGBench tasks, and Figures 18 and 19 show the per-task offline-to-online learning curves.

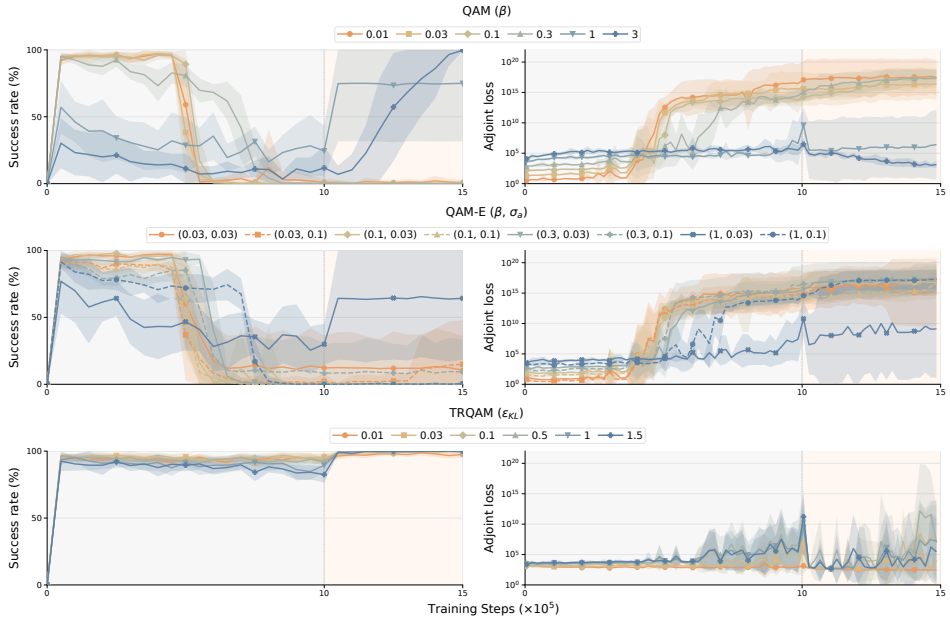


Figure 11: **Hyperparameter sweep on** `Robomimic-lift` **(8 seeds).** Sweep ranges for QAM, QAM-E, and TRQAM follow Table 6. The left column reports success rate, and the right column reports adjoint-matching loss on log scale. Fixed-temperature methods (QAM, QAM-E) exhibit adjoint loss growth of 10 to 20 orders of magnitude across most settings, with corresponding success rate collapse. TRQAM remains stable across all six budgets. Shaded regions denote ± 1 standard deviation across seeds.

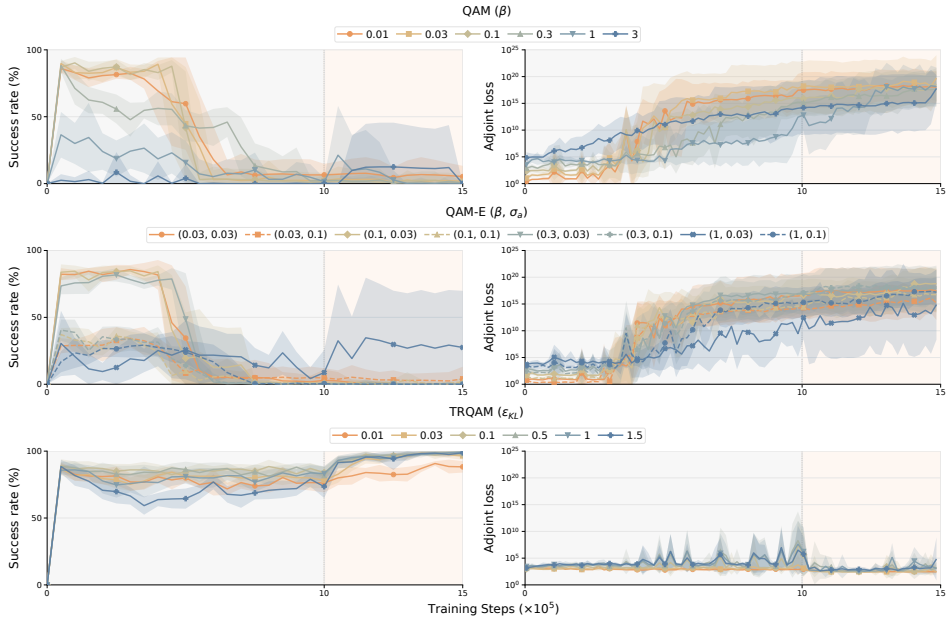


Figure 12: **Hyperparameter sweep on** `Robomimic-can` **(8 seeds).** Same setup as Figure 11. Fixed-temperature collapse is again hyperparameter-wide. QAM and QAM-E exhibit adjoint loss growth of 10 to 25 orders of magnitude across most settings with success rate collapse, while TRQAM remains stable across all six budgets. Shaded regions denote ± 1 standard deviation across seeds.

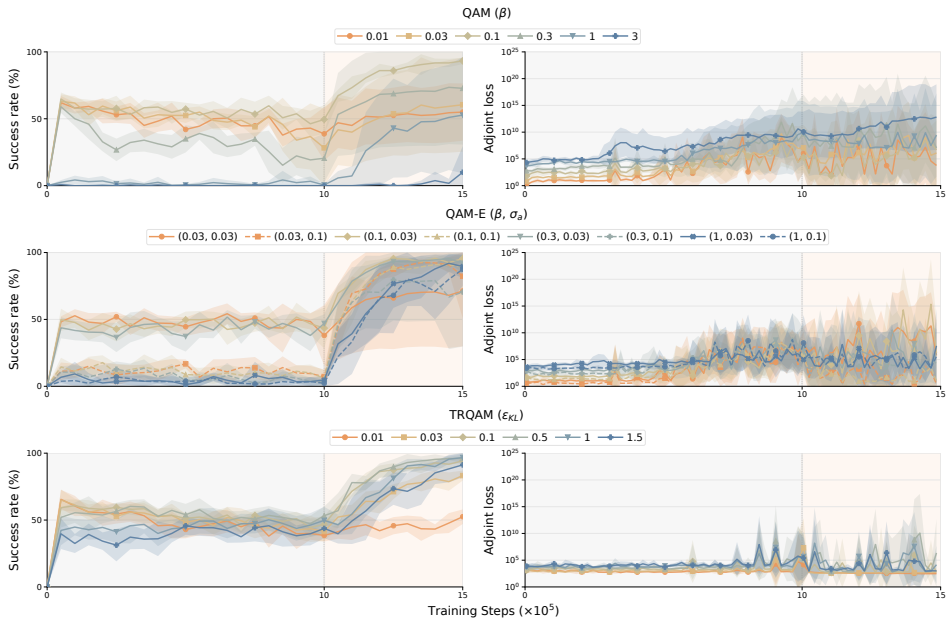


Figure 13: **Hyperparameter sweep on `Robomimic-square` (8 seeds).** Same setup as Figure 11. While the adjoint loss explosion is less severe than on `lift` and `can`, fixed-temperature variants still exhibit signs of instability across the sweep, with adjoint loss steadily growing during offline training. TRQAM, in contrast, maintains a bounded adjoint loss and matches or exceeds the fixed-temperature variants across the sweep. Shaded regions denote ± 1 standard deviation across seeds.

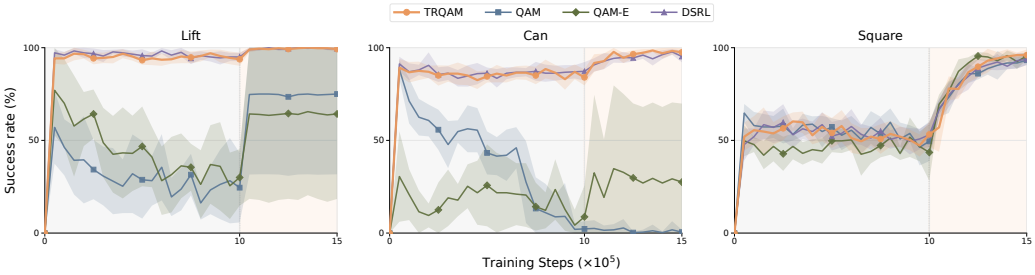


Figure 14: **Best per-method hyperparameters on Robomimic across all three tasks (8 seeds).** For each method (TRQAM, QAM, QAM-E, DSRL) we select the configuration with the strongest overall offline-to-online learning curve from the sweep in Table 6. Shaded regions denote ± 1 standard deviation across seeds.

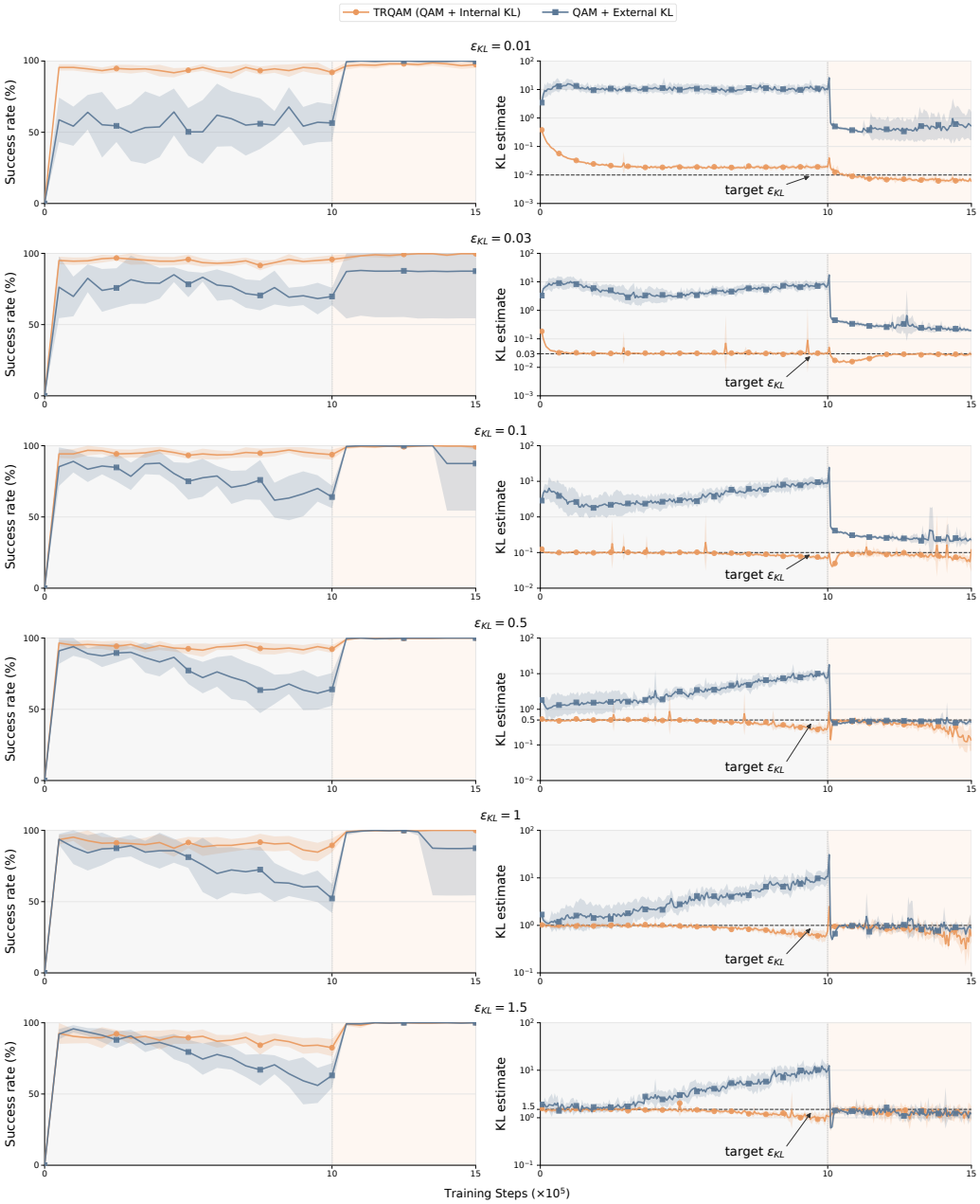


Figure 15: **Internal vs. external KL regularization on `Robomimic-lift` across all budgets (8 seeds).** Each row plots one target KL budget $\varepsilon_{\mathrm{KL}} \in \{0.01, 0.03, 0.1, 0.5, 1.0, 1.5\}$. The left column shows success rate over training steps, and the right column shows realized path-space KL with the target shown as a dashed line. TRQAM (orange) tracks each prescribed budget tightly across offline and online training, whereas QAM with external KL regularization (blue) lets the realized KL drift well over it, with corresponding success rate degradation. Shaded regions denote ± 1 standard deviation across seeds.

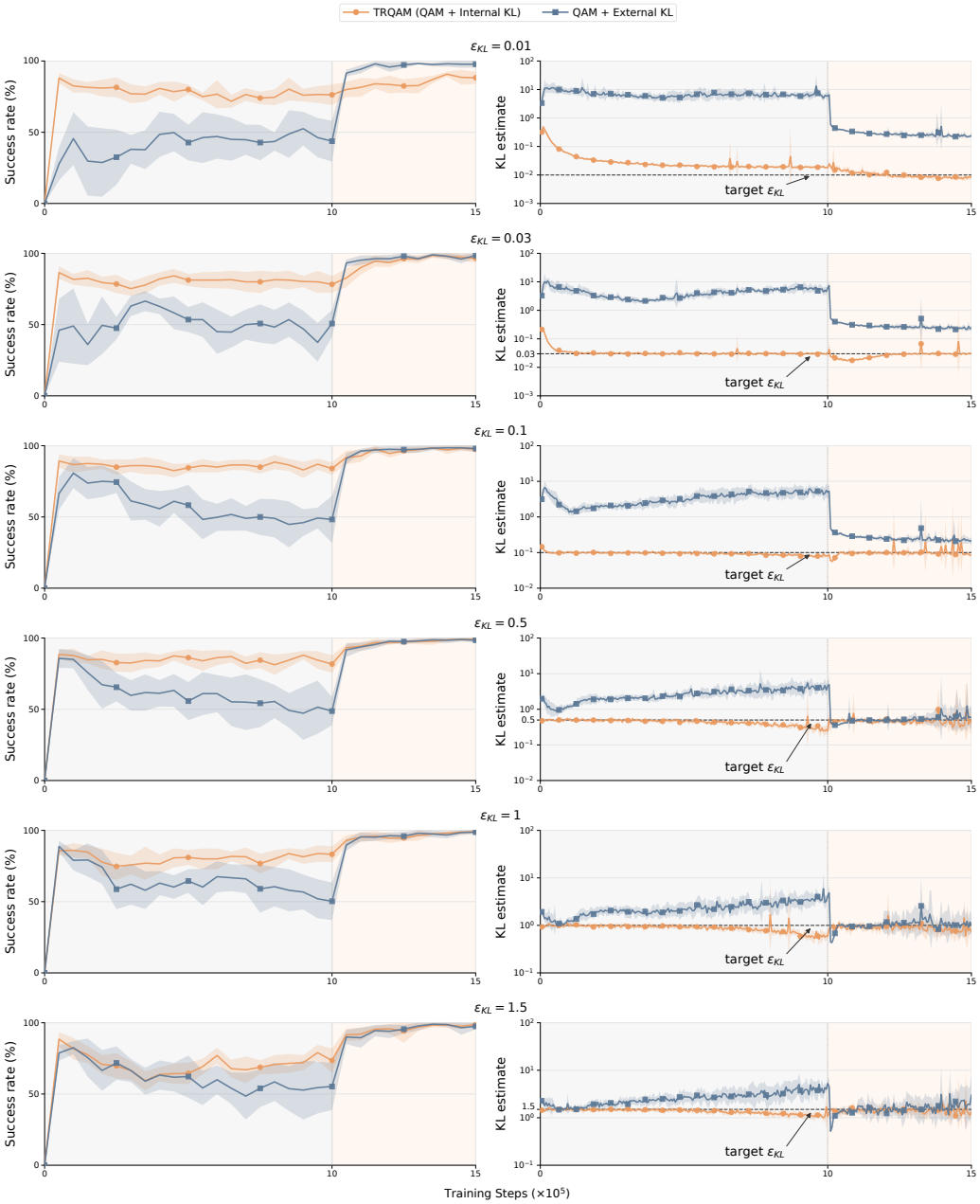


Figure 16: **Internal vs. external KL regularization on `Robomimic-can` across all budgets (8 seeds).** Each row plots one target KL budget $\varepsilon_{\mathrm{KL}} \in \{0.01, 0.03, 0.1, 0.5, 1.0, 1.5\}$. The left column shows success rate over training steps, and the right column shows realized path-space KL with the target shown as a dashed line. TRQAM (orange) tracks each prescribed budget tightly across offline and online training, whereas QAM with external KL regularization (blue) lets the realized KL drift well over it, with corresponding success rate degradation. Shaded regions denote ± 1 standard deviation across seeds.

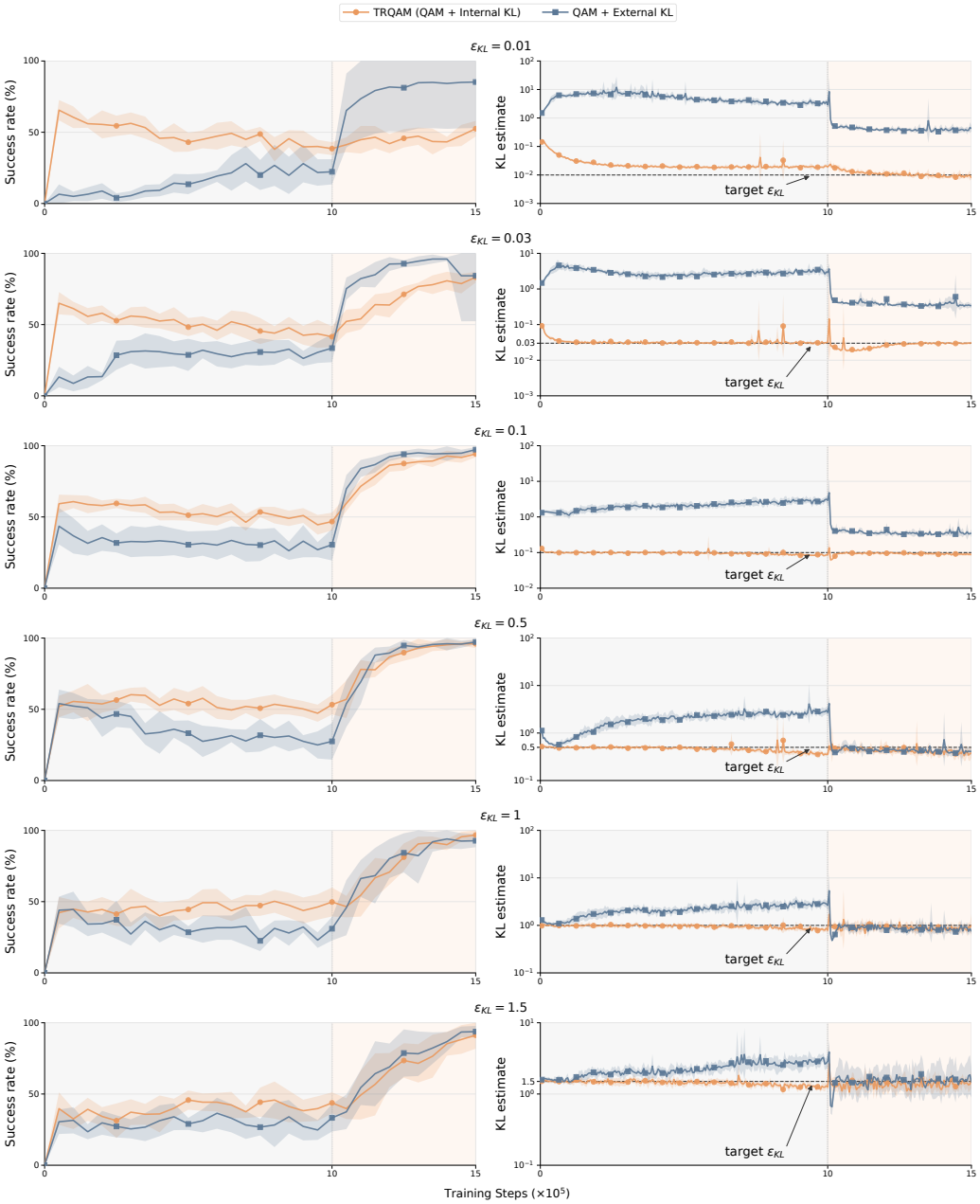


Figure 17: **Internal vs. external KL regularization on `Robomimic-square` across all budgets (8 seeds).** Each row plots one target KL budget $\varepsilon_{\mathrm{KL}} \in \{0.01, 0.03, 0.1, 0.5, 1.0, 1.5\}$. The left column shows success rate over training steps, and the right column shows realized path-space KL with the target shown as a dashed line. TRQAM (orange) tracks each prescribed budget tightly across offline and online training, whereas QAM with external KL regularization (blue) lets the realized KL drift well over it, with corresponding success rate degradation. Shaded regions denote ± 1 standard deviation across seeds.

Table 16: **Full offline results at 1M training steps (8 seeds).** Mean success rate (%) ± one standard deviation.

			FQL	CGQL-L	DSRL	IFQL	QAM	QAM-E	TRQAM
antmaze-large	task1		24 ±44	62 ±38	45 ±12	7 ±11	96 ±3	93 ±5	**95** ±4
	task2		0 ±0	21 ±36	73 ±7	9 ±14	36 ±49	91 ±4	**87** ±6
	task3		73 ±14	88 ±5	84 ±5	62 ±24	86 ±7	93 ±3	**93** ±3
	task4		0 ±0	0 ±0	0 ±0	35 ±22	0 ±0	66 ±9	**75** ±17
	task5		92 ±4	68 ±42	63 ±8	34 ±24	94 ±3	89 ±5	**96** ±3
	agg.	(5 tasks)	38 ±9	48 ±7	53 ±2	29 ±8	62 ±9	86 ±3	**89** ±4
antmaze-giant-10M	task1		1 ±1	11 ±12	2 ±3	0 ±0	37 ±9	0 ±0	**48** ±5
	task2		0 ±0	14 ±12	0 ±0	43 ±13	18 ±19	0 ±0	21 ±12
	task3		0 ±0	0 ±0	0 ±0	14 ±8	2 ±2	0 ±0	5 ±5
	task4		0 ±0	0 ±0	0 ±0	1 ±2	20 ±24	0 ±0	**58** ±13
	task5		10 ±29	8 ±23	2 ±2	2 ±2	69 ±8	28 ±38	**75** ±7
	agg.	(5 tasks)	2 ±6	7 ±5	1 ±1	12 ±3	29 ±4	6 ±8	**41** ±4
humanoidmaze-medium	task1		84 ±21	84 ±9	24 ±23	93 ±3	33 ±34	28 ±30	**87** ±5
	task2		99 ±1	99 ±1	88 ±6	94 ±4	100 ±1	98 ±2	89 ±7
	task3		89 ±9	0 ±0	64 ±27	96 ±4	90 ±8	74 ±18	**90** ±6
	task4		0 ±0	0 ±0	0 ±0	82 ±8	0 ±0	0 ±0	61 ±7
	task5		99 ±1	100 ±1	88 ±5	99 ±2	100 ±1	99 ±1	**95** ±4
	agg.	(5 tasks)	74 ±5	57 ±2	53 ±10	93 ±2	64 ±7	60 ±6	84 ±3
humanoidmaze-large	task1		1 ±2	16 ±11	1 ±1	32 ±5	6 ±10	9 ±14	**45** ±8
	task2		0 ±0	0 ±0	0 ±0	6 ±9	0 ±0	0 ±0	**10** ±6
	task3		7 ±4	16 ±8	2 ±3	76 ±6	7 ±5	3 ±4	34 ±15
	task4		0 ±0	0 ±1	0 ±0	35 ±26	0 ±0	0 ±0	**44** ±13
	task5		0 ±1	0 ±0	1 ±1	0 ±0	5 ±11	9 ±18	**45** ±6
	agg.	(5 tasks)	2 ±1	6 ±3	1 ±1	30 ±7	4 ±3	4 ±5	**36** ±4
scene	task1		100 ±0	100 ±0	100 ±0	99 ±1	100 ±0	100 ±0	**100** ±0
	task2		100 ±1	99 ±1	100 ±0	2 ±2	100 ±0	100 ±0	**100** ±0
	task3		82 ±5	93 ±7	100 ±1	77 ±7	94 ±4	93 ±4	**100** ±1
	task4		60 ±22	0 ±0	100 ±0	2 ±2	26 ±21	22 ±30	**93** ±6
	task5		8 ±7	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±1
	agg.	(5 tasks)	70 ±5	58 ±1	80 ±0	36 ±1	64 ±4	63 ±6	**79** ±1
puzzle-3x3	task1		87 ±11	0 ±0	100 ±0	100 ±1	75 ±15	99 ±2	**100** ±0
	task2		37 ±39	0 ±0	100 ±0	23 ±7	0 ±0	100 ±0	**100** ±0
	task3		0 ±0	0 ±0	100 ±0	82 ±11	0 ±0	76 ±11	**100** ±0
	task4		0 ±0	0 ±0	100 ±0	29 ±8	0 ±0	87 ±6	**100** ±1
	task5		3 ±6	0 ±0	100 ±0	86 ±9	0 ±1	84 ±20	**100** ±0
	agg.	(5 tasks)	25 ±10	0 ±0	100 ±0	64 ±4	15 ±3	89 ±4	**100** ±0
puzzle-4x4-10M	task1		3 ±3	1 ±1	90 ±5	74 ±11	3 ±4	90 ±10	**100** ±0
	task2		6 ±9	0 ±0	9 ±9	4 ±3	1 ±1	43 ±22	**99** ±1
	task3		23 ±15	0 ±0	81 ±20	84 ±6	2 ±3	57 ±10	**98** ±2
	task4		7 ±5	0 ±0	76 ±28	46 ±13	0 ±1	31 ±16	**100** ±1
	task5		9 ±16	0 ±0	49 ±43	3 ±1	1 ±1	50 ±22	**98** ±5
	agg.	(5 tasks)	9 ±7	0 ±0	61 ±8	42 ±4	1 ±1	54 ±8	**99** ±1
cube-double	task1		75 ±10	83 ±4	85 ±7	18 ±7	96 ±3	94 ±4	**98** ±2
	task2		59 ±11	66 ±6	85 ±5	9 ±2	81 ±6	82 ±7	**92** ±3
	task3		39 ±8	72 ±7	82 ±10	8 ±4	75 ±7	67 ±8	**80** ±12
	task4		6 ±3	15 ±5	35 ±6	2 ±2	24 ±6	29 ±6	**54** ±8
	task5		44 ±9	37 ±6	71 ±4	7 ±3	78 ±6	82 ±6	**82** ±6
	agg.	(5 tasks)	44 ±4	55 ±2	72 ±4	9 ±2	71 ±2	71 ±3	**81** ±3
cube-triple-10M	task1		26 ±20	2 ±3	70 ±18	51 ±22	56 ±22	26 ±11	**87** ±8
	task2		1 ±1	0 ±0	16 ±7	16 ±3	15 ±11	14 ±9	**42** ±9
	task3		8 ±6	0 ±0	40 ±8	28 ±8	21 ±11	9 ±6	**45** ±9
	task4		2 ±3	0 ±0	15 ±5	1 ±1	4 ±3	3 ±3	**30** ±13
	task5		1 ±1	0 ±0	31 ±12	23 ±8	1 ±1	2 ±4	**48** ±17
	agg.	(5 tasks)	7 ±5	0 ±1	34 ±6	24 ±7	19 ±6	11 ±4	**50** ±5
cube-quadruple-100M	task1		36 ±25	7 ±5	36 ±12	15 ±11	70 ±10	45 ±19	**66** ±16
	task2		4 ±5	0 ±1	5 ±3	5 ±7	2 ±4	0 ±0	**18** ±11
	task3		3 ±3	0 ±0	2 ±2	6 ±8	16 ±11	2 ±4	**11** ±7
	task4		0 ±0	0 ±0	0 ±1	0 ±0	2 ±2	0 ±0	**3** ±2
	task5		0 ±0	0 ±0	1 ±1	2 ±3	0 ±0	0 ±0	0 ±0
	agg.	(5 tasks)	9 ±5	1 ±1	9 ±3	6 ±3	18 ±3	9 ±3	**19** ±5
all	agg.	(50 tasks)	28	23	46	35	35	45	**68**

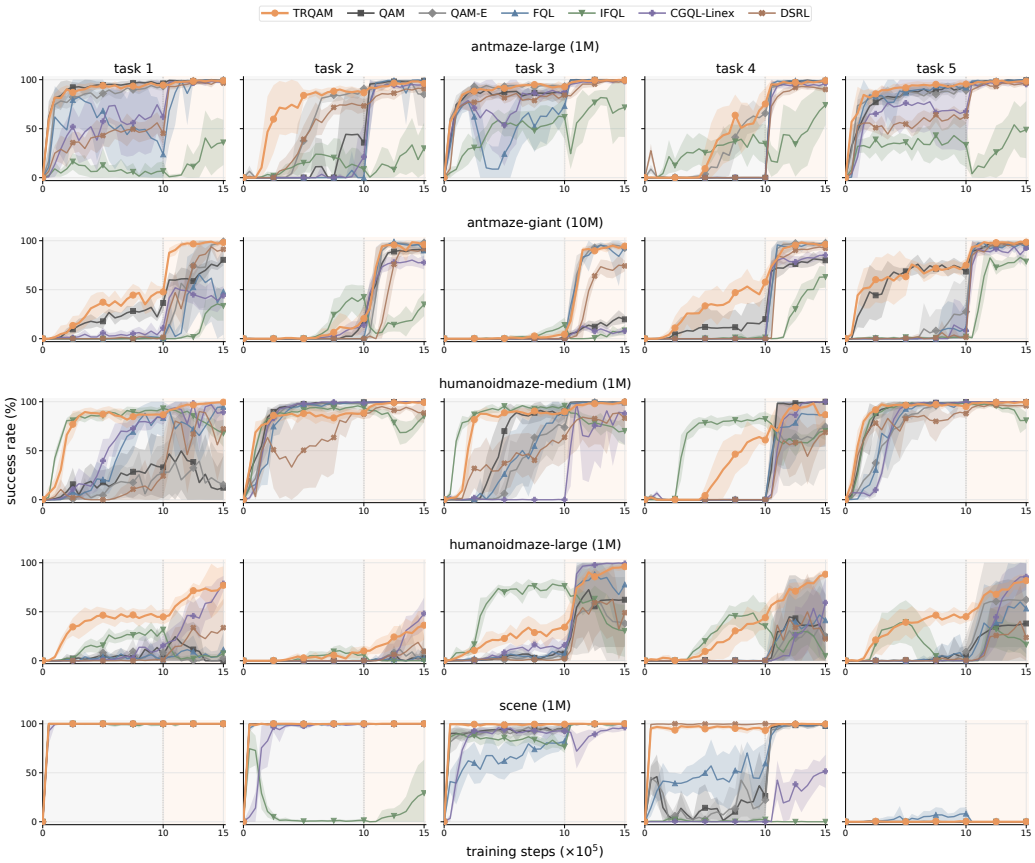


Figure 18: **Per-task offline-to-online learning curves on OGBench [40] (8 seeds).** The domains are `antmaze-large`, `antmaze-giant`, `humanoidmaze-medium`, `humanoidmaze-large`, and `scene`. Each panel reports success rate over training steps for all seven methods (TRQAM, QAM, QAM-E, FQL, IFQL, CGQL-Linex, DSRL). The offline-to-online transition occurs at 10^6 steps. Shaded regions denote ± 1 standard deviation across seeds.

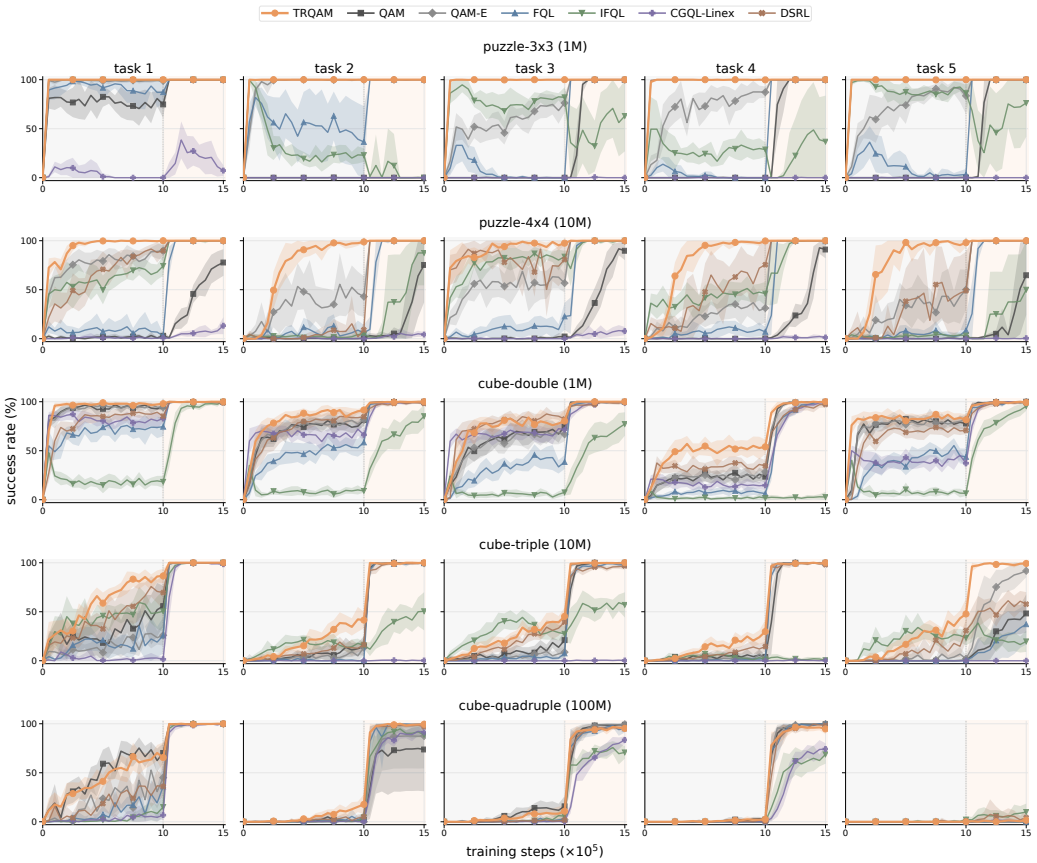


Figure 19: **Per-task offline-to-online learning curves on OGBench [40] (8 seeds).** The domains are `puzzle-3x3`, `puzzle-4x4`, `cube-double`, `cube-triple`, and `cube-quadruple`. Each panel reports success rate over training steps for all seven methods (TRQAM, QAM, QAM-E, FQL, IFQL, CGQL-Linex, DSRL). The offline-to-online transition occurs at 10^6 steps. Shaded regions denote ± 1 standard deviation across seeds.

J Broader impacts

TRQAM is a method for stable off-policy fine-tuning of pretrained flow policies, evaluated only on simulated benchmarks. More reliable fine-tuning can reduce the data and compute needed to adapt pretrained robot policies to new tasks. However, a KL bound on the policy does not rule out unsafe behavior once such policies are deployed in the real world, so safety filters and careful evaluation remain necessary. We do not release models or datasets that pose high misuse risk.

K Limitations and future work

Computational cost. Computing the Adjoint Matching loss requires a vector-Jacobian product (VJP) through the velocity field at each step of the backward ODE. This cost grows with the model size. Its share of per-step training time does not grow without bound, but it reaches about a fifth for larger actors in our measurements (Appendix H.9), so it is hard to ignore for large policies. Scaling TRQAM to substantially larger pretrained policies, such as vision-language-action models with visual inputs, remains to be tested.

Dependence on the pretrained policy. TRQAM keeps the fine-tuned policy close to π_{base} by design. Its performance is therefore tied to the quality of the pretrained policy, and a poor prior limits what any trust-region method can recover.

Memoryless OT schedule. Our closed-form per-step KL estimator relies on the memoryless OT schedule, under which the discretized transitions are Gaussian with a shared covariance. Other schedules whose Euler transitions share this structure admit an analogous estimator, but the exact expressions change, and we leave them to future work.

Finite-sample budget tracking. Theorem 1 is an exact identity for a given control, and the discretized sampler admits a closed-form per-step KL. In practice, however, TRQAM tracks the budget with a Monte Carlo estimate smoothed by an exponential moving average, while the critic, the policy, and λ are updated simultaneously. The realized KL therefore stays close to $\varepsilon_{\mathrm{KL}}$ but does not satisfy the budget exactly at every iteration.

Choice of the KL budget. $\varepsilon_{\mathrm{KL}}$ remains a hyperparameter. A small budget works well on most domains, but domains with large state spaces such as `puzzle-4x4` benefit from larger budgets (Appendix H.3). Because TRQAM tracks time-varying budgets (Appendix H.2), adjusting $\varepsilon_{\mathrm{KL}}$ automatically from a training signal is a natural direction for future work.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Section 1 state three contributions, namely identification of critic-error amplification (Lemma 1), the Girsanov identity that makes λ the exact weight of the path-space KL in the SOC objective (Theorem 1), and the TRQAM algorithm enforcing a target KL bound through projected dual descent, which are formalized in Sections 3 to 4 and validated by the offline and offline-to-online RL results on 50 OGBench tasks in Section 4.1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix K discusses the limitations of TRQAM, including the per-step vector-Jacobian product cost of Adjoint Matching, the dependence on the pretrained policy, the reliance on the memoryless OT schedule, finite-sample budget tracking, and the choice of $\varepsilon_{\mathrm{KL}}$.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best

judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: Theorem 1 (path-space KL identity), Proposition 1 (terminal KL bounded by path-space KL), Proposition 2 (tilted-policy KL is non-decreasing in β), and Lemma 1 (exponential amplification of critic errors) are stated in Section 3, and the full assumptions and complete proofs are provided in Appendix D (Appendices D.1 to D.4).

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Appendix C reports the benchmarks (50 OGBench tasks and Robomimic), per-domain dataset sizes and episode metadata (Appendix C.1), 8 evaluation seeds per setting, and the offline-to-online evaluation protocol. Appendix C.3 gives the per-method hyperparameter sweep ranges and selected values, and algorithm pseudocode is provided in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.

(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The code is publicly available at https://github.com/yonghdong/trqam. The paper also provides full algorithmic pseudocode (Appendix A), detailed hyperparameter and protocol descriptions (Appendix C), and uses publicly available benchmarks (OGBench, Robomimic). For `antmaze-giant`, whose 10M-transition dataset is not publicly released, we generate the data with the official OGBench data-generation pipeline and release it at https://huggingface.co/datasets/yonghoon96/antmaze-giant-navigate-10m-v0 (Appendix C.1). The behavior-cloning checkpoints used to initialize all methods are available at https://huggingface.co/yonghoon96/trqam-bc-checkpoints.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendix C.3 report optimizer settings, learning rates, network architectures, batch sizes, replay-buffer composition, training/evaluation splits, the 8 evaluation seeds, and method-specific hyperparameters in Tables 4 to 6. The per-method tuning protocol (sweep configurations and selection criterion) is described in Appendix C.3.

   Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All main results are averaged over 8 random seeds. Tables report mean and standard deviation across seeds, and learning curves show the mean with shaded ± 1 standard-deviation regions across seeds (Section 4 and Appendix H).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix C.2 reports the GPU types (NVIDIA RTX 2080 Ti and A100-80GB, with a small share on RTX 3090, A6000, and 4090), the median wallclock time per run by dataset scale and GPU (Table 2), and the total compute of approximately 38,000 GPU-hours for the reported experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: The work uses only publicly released simulated benchmarks (OGBench, Robomimic), does not involve human subjects, personal data, or ethically sensitive collection, and conforms to the NeurIPS Code of Ethics.

Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Broader impacts of the proposed off-policy fine-tuning method, including positive impacts on data- and compute-efficient policy fine-tuning and potential negative impacts under deployment beyond simulated benchmarks, are discussed in Appendix J.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The released assets are behavior-cloning flow policies and a dataset for simulated control benchmarks, which pose no high misuse risk. The paper studies an off-policy RL fine-tuning algorithm evaluated on simulated robotics benchmarks.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing assets are properly cited and used within their licenses: OGBench [40] (MIT License) and Robomimic [32] (MIT License). The `antmaze-giant` dataset is generated with the official OGBench data-generation pipeline under its default settings, since its 10M-transition version is not publicly released (Appendix C.1).

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We release the code at `https://github.com/yonghdong/trqam` under the MIT License, with a README describing installation, training, and evaluation. We also release the generated 10M-transition `antmaze-giant` dataset at `https://huggingface.co/datasets/yonghoon96/antmaze-giant-navigate-10m-v0` under the MIT License, with a dataset card, and the behavior-cloning checkpoints for all ten OGBench domains at `https://huggingface.co/yonghoon96/trqam-bc-checkpoints` under the MIT License, with a model card.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects, so IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: Large language models are not used as a component of the core methodology. TRQAM concerns off-policy reinforcement learning fine-tuning of flow-matching policies for continuous-control tasks.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.