Q-LEARNING WITH SCALAR ADJOINT MATCHING

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow policies capture rich and diverse action distributions, and fine-tuning them with off-policy RL to improve beyond the demonstrations has drawn growing interest. However, fine-tuning a flow policy against a learned value function is not trivial, because the policy generates its action over many flow steps. Adjoint matching offers a principled way to update the flow model itself by propagating value information from the final action back to each flow step, but it requires a vector–Jacobian product through the policy at every step, a cost that grows with the number of flow steps and the policy size. We observe that the batch-averaged velocity Jacobian of pretrained flow policies concentrates on its diagonal. Motivated by this finding, we derive a closed-form scalar adjoint that scales the value gradient at the final action by the flow time, eliminating the per-step vector–Jacobian products. We further find that controlling the critic's value at policy-generated actions is particularly important under the scalar adjoint. Based on these findings, we propose Q-learning with Scalar Adjoint Matching (SQAM), which combines the scalar adjoint with a value penalty at those actions. SQAM's gains concentrate on the four hardest OGBench domains, where its success rate exceeds that of the strongest baseline in each domain by 18 to 35 percentage points. To test whether SQAM extends to large pretrained policies, we also fine-tune a vision-language-action policy on a real bimanual robot. SQAM improves over supervised fine-tuning on all three tasks. Videos are available at https://sqam-iclr2027.github.io/.

1 INTRODUCTION

Flow policies represent rich and multimodal action distributions, and they have become a common policy class for offline and offline-to-online RL (Park et al., 2025b; Li & Levine, 2026; Dong et al., 2026c; Chi et al., 2023) as well as the backbone of recent large pretrained robot policies (Intelligence et al., 2025b; Bjorck et al., 2025). Improving such a policy beyond the data it was trained on calls for fine-tuning it with off-policy RL against a learned action-value function (Intelligence et al., 2025a; Wang et al., 2026; Dong et al., 2026a; Xu et al., 2026; Li et al., 2025b; Chen et al., 2025).

However, fine-tuning a flow policy to maximize a learned value function is not a trivial problem because of its multiple sampling procedure (Park et al., 2025b; Zhang et al., 2026). Most methods therefore freeze the policy and work outside it, adding a separately learned residual to the actions it produces (Dong et al., 2026b; Xiao et al., 2026) or learning with RL which input noise to feed the frozen policy (Wagenmaker et al., 2025), which leaves the pretrained policy intact during training. The improvement then rests on a separately learned component rather than on the generative model itself, so it is either bounded by what the frozen policy can express or forfeits the expressiveness of the flow policy. On the other hand, adjoint matching (Domingo-Enrich et al., 2025; Li & Levine, 2026; Dong et al., 2026c) provides a theoretically grounded alternative for directly updating the flow model with value functions. It propagates value information from the final action back to each flow step to guide the model's updates. However, this requires a vector–Jacobian product through the policy at every step, so the computational cost grows with the number of flow steps and policy size.

In this work, we observe that the batch-averaged velocity Jacobian of a pretrained flow policy concentrates on its diagonal (Figure 2). This suggests that the vector–Jacobian product adjoint matching pays for at every flow step may not be necessary. Inspired by this finding, we approximate the Jacobian by a scalar multiple of the identity, under which the propagation admits a closed form that scales the value gradient at the final action by the flow time (Proposition 1). We call this the scalar adjoint.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

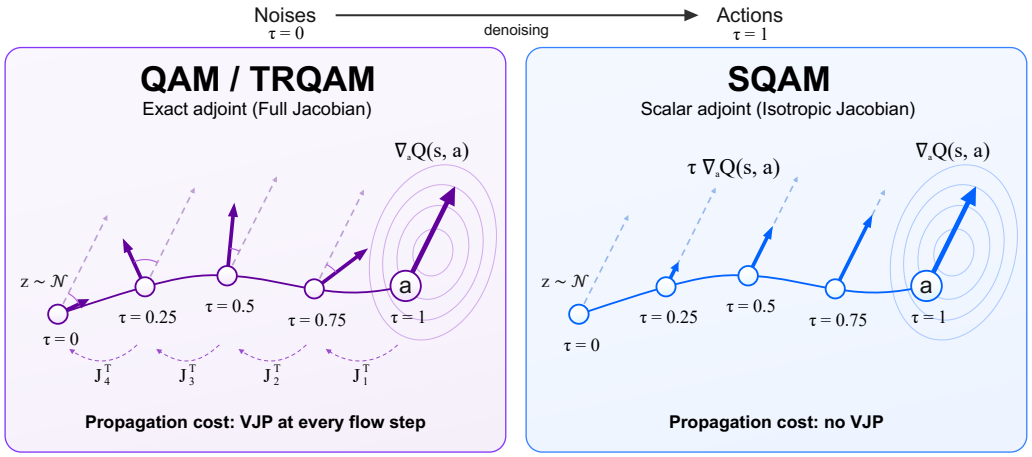


Figure 1: Adjoint matching propagates the critic's gradient at the final action back to the flow steps that produced it, and the propagated gradient updates the flow model. **Left:** QAM and TRQAM integrate the lean adjoint ODE backward along the sampling chain, which takes a vector–Jacobian product $J_i^\top$ through the policy at every flow step, so the cost grows with the number of flow steps and the policy size. **Right:** SQAM instead scales the gradient at the final action, $\nabla_a Q(s, a)$, by the flow time τ, with no vector–Jacobian product at any flow step.

It needs a single critic gradient at the sampled action and no vector–Jacobian product through the policy, and Section 4.2 shows that it preserves the qualitative trends of the exact propagation.

We further find that the effectiveness of the scalar adjoint depends on the value-learning scheme. We investigate this dependence and identify the critic's value at policy-generated actions as particularly important. Our regularization ablations isolate this dependency. Methods that regulate the critic at these actions improve performance, while methods that act only on dataset (Kostrikov et al., 2022) or action distance (Tarasov et al., 2023) provide little improvement (Section 4.3). Based on these findings, we propose *Q-learning with Scalar Adjoint Matching* (SQAM), which combines the scalar adjoint with a value penalty at policy-generated actions.

Through experiments on the 50 OGBench (Park et al., 2025a) tasks, SQAM outperforms prior methods in both offline RL and offline-to-online RL, and the gains concentrate where the benchmark is hardest. On the four hardest domains it adds 18 to 35 percentage points of offline success rate over the best baseline in each, and it reaches 75% against 68% for the strongest baseline overall. To assess whether SQAM extends to large pretrained policies, we apply it to offline-to-online RL fine-tuning of a vision-language-action model (Kim et al., 2026b) on a real-world bimanual robot (Rainbow Robotics, 2024) equipped with two 20-DoF dexterous hands (Wuji Technology, 2025). We observe that SQAM clearly improves over supervised fine-tuning on all three manipulation tasks, while residual fine-tuning (Dong et al., 2026a) struggles in the same setting.

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

- We introduce SQAM, an off-policy fine-tuning algorithm for flow policies that reduces the compute cost of adjoint matching by replacing its per-step vector–Jacobian products with the closed-form scalar adjoint.
- We show that the scalar adjoint makes performance depend on the value learning scheme, and we find that what matters is controlling the critic's value at the actions the current policy generates.
- SQAM's gains concentrate on the four hardest OGBench domains, where it exceeds the strongest baseline by 18 to 35 percentage points in success rate, and it extends to fine-tuning a pretrained vision-language-action policy on a real robot.

2 Background

Flow matching and flow policies. Flow matching trains a velocity field $v_\theta(x, \tau)$ that transports samples from a source distribution p_0 to a target distribution p_1 through $\mathrm{d}X_\tau = v_\theta(X_\tau, \tau)\,\mathrm{d}\tau$

(Lipman et al., 2023; Albergo et al., 2023; Liu et al., 2023). We consider the commonly used linear interpolation path $X_\tau = (1-\tau)X_0 + \tau X_1$, for which the training objective is

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

A flow policy applies this to action generation. Given a state s, an action is drawn by integrating $\mathrm{d}X_\tau = v_\theta(s, X_\tau, \tau)\,\mathrm{d}\tau$ from $X_0 \sim \mathcal{N}(0, I)$ and taking the endpoint X_1 (Park et al., 2025b).

Reinforcement learning. We consider a Markov decision process (Sutton & Barto, 2018) $(\mathcal{S}, \mathcal{A}, P, r, \gamma, \rho_0)$ with state space $\mathcal{S}$, action space $\mathcal{A}$, transition kernel $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, reward $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, initial state distribution $\rho_0 \in \Delta(\mathcal{S})$, and discount $\gamma \in [0, 1)$, where $\Delta(\cdot)$ is the set of probability distributions over a set. The goal is to learn a policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ maximising the expected discounted return $\mathbb{E}_{\rho_0,\pi,P}\big[\sum_{t\geq 0}\gamma^t r(s_t, a_t)\big]$. We follow the actor-critic approach, which learns an action-value function $Q^\pi(s,a) = \mathbb{E}_\pi\big[\sum_{k\geq 0}\gamma^k r(s_{t+k}, a_{t+k})|s_t = s, a_t = a\big]$, also called the critic, and improves the policy by maximizing it. Our setting is off-policy fine-tuning of a pretrained flow policy. Training draws transitions (s, a, r, s') from a replay buffer $\mathcal{D}$, which initially holds a dataset collected by a behavior policy that may be unknown or suboptimal, and on which the flow policy $\pi_{\mathrm{base}}(\cdot|s)$ is pretrained by behavior cloning. Fine-tuning first runs offline on that data alone, and then online, where $\mathcal{D}$ grows with the rollouts the fine-tuned policy collects.

Adjoint matching. A flow policy generates its action over many flow steps, and the critic only tells us how good the final action is. To improve the flow policy, each intermediate step also needs to know which direction to push its state so that the final action gets a higher value. Adjoint matching (Domingo-Enrich et al., 2025; Li & Levine, 2026) computes this direction for every step. Starting from the critic's gradient at the final action, it passes that gradient back through the chain one step at a time by integrating the lean adjoint ODE,

$$\mathrm{d}\tilde{a}_\tau = -\tilde{a}_\tau^\top \nabla_{X_\tau}\Big(2v^{\mathrm{base}}(X_\tau,\tau) - \frac{1}{\tau}X_\tau\Big)\,\mathrm{d}\tau, \qquad \tilde{a}_1 = -\nabla_{X_1}Q^\pi(s, X_1), \tag{1}$$

and trains the fine-tuned velocity v_θ^{ft} to match the gradient that Equation (1) delivers to each step,

$$\mathcal{L}_{\mathrm{Adj-Match}}(\theta) \;=\; \mathbb{E}\Big[\sum_\tau \Big\|\frac{2}{\sigma(\tau)}\big(v_\theta^{\mathrm{ft}}(X_\tau,\tau) - v^{\mathrm{base}}(X_\tau,\tau)\big) + \sigma(\tau)\tilde{a}_\tau\Big\|^2\Big], \tag{2}$$

where $\sigma(\tau)$ is the noise schedule of the sampler. Dong et al. (2026c) stabilize the optimization by constraining the fine-tuned sampler to a path-space KL budget $\varepsilon_{\mathrm{KL}}$ (Section B).

3 SQAM: Q-Learning with Scalar Adjoint Matching

3.1 A closed-form scalar adjoint under an isotropic velocity Jacobian

Solving the lean adjoint ODE in Equation (1) requires a vector–Jacobian product with the velocity Jacobian $\nabla_{X_\tau} v^{\mathrm{base}}(X_\tau,\tau)$ at every flow step, so the cost of each update grows with both the number of flow steps and the policy size, which makes adjoint matching difficult to scale as pretrained flow policies grow larger (Intelligence et al., 2025b; Bjorck et al., 2025). We resolve this problem with the empirical finding that the batch-averaged velocity Jacobian $J_\tau := \nabla_{X_\tau} v^{\mathrm{base}}(X_\tau,\tau)$ of the pretrained policy concentrates on its diagonal (Figure 2, with all ten domains in Section F.4). Following the isotropic approximation of the posterior covariance in Peng et al. (2024), we approximate $J_\tau = c_\tau I$ with a scalar c_τ, which solves Equation (1) in closed form.

Proposition 1 (the lean adjoint ODE under an isotropic velocity Jacobian)**.** *Assume $J_\tau = c_\tau I$ for a scalar c_τ. Then the lean adjoint ODE in Equation (1) admits the closed-form solution*

$$\tilde{a}_\tau \;=\; -\tau\,\exp\Big(2\int_\tau^1 c_s\,\mathrm{d}s\Big)\,\nabla_{X_1}Q^\pi(s, X_1),$$

which points along the negative of the critic's action gradient at the final action at every flow time $\tau > 0$.

Proof. With $J_\tau = c_\tau I$, the Jacobian term in Equation (1) becomes $\big(\frac{1}{\tau} - 2c_\tau\big)I$, so the ODE reads $\mathrm{d}\tilde{a}_\tau = \big(\frac{1}{\tau} - 2c_\tau\big)\tilde{a}_\tau\,\mathrm{d}\tau$ and every coordinate is scaled by the same factor. Integrating from τ to 1 gives $\tilde{a}_\tau = \tau\exp\big(2\int_\tau^1 c_s\,\mathrm{d}s\big)\tilde{a}_1$, and the terminal condition is $\tilde{a}_1 = -\nabla_{X_1}Q^\pi(s, X_1)$. □

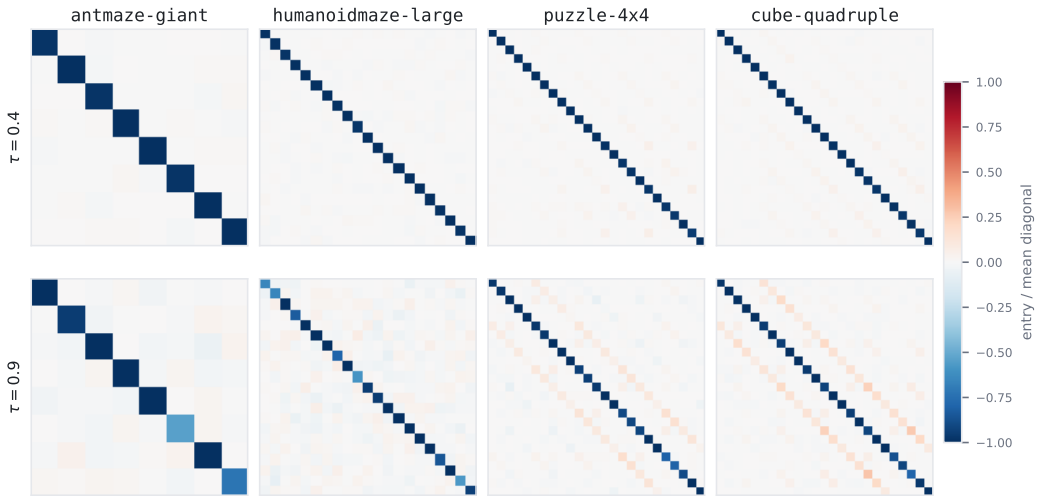


Figure 2: Velocity Jacobian of the pretrained flow on four domains at two flow times, averaged over a batch of states and normalized per panel by the mean diagonal. The batch-averaged Jacobian is dominated by its diagonal.

At every flow time, the closed-form solution is a scalar multiple of the critic's action gradient at the final action. In practice, estimating c_τ would re-introduce the Jacobian we set out to avoid, so we eliminate the exponential factor and keep only

$$\hat{a}_\tau = -\tau \nabla_{X_1} Q^\pi(s, X_1), \qquad \forall \tau \in [0, 1],$$

which we call *scalar adjoint*. The exponential factor $\exp\left(2\int_\tau^1 c_s \, \mathrm{d}s\right)$ only rescales the closed-form solution, and dropping it is empirically justified by Figure 4 (left), where the norm of $\hat{a}_\tau$ follows the same trend as that of $\tilde{a}_\tau$ across flow times. The scalar adjoint requires a single critic gradient at the sampled action X_1, and no vector–Jacobian product through the policy at any flow step.

Moreover, SQAM follows the adjoint matching form of Equation (2), replacing the exact adjoint $\tilde{a}_\tau$ in its target with the scalar adjoint $\hat{a}_\tau$, so the path-space trust region of TRQAM (Dong et al., 2026c) can be used or left out. We use it by default for stable training, and Section F.3 reports SQAM without it. Section F.1 compares the computational cost of the scalar and exact adjoint updates.

3.2 Regularizing the critic at the policy's own actions

Using the scalar adjoint alone does not work well, especially on the manipulation domains (Section 4.3). The exact adjoint passes $\nabla_{X_1} Q^\pi(s, X_1)$ through the velocity Jacobian at every flow step, so it does not use this gradient as directly as the scalar adjoint does (Figure 1). The scalar adjoint update therefore depends directly on the critic's gradient at the policy-generated action, and we hypothesize that its failure arises from errors in the critic at that action. To address this, we introduce a value penalty on the critic at the actions the current policy generates, relative to the value at the dataset action,

$$\mathcal{L}_{\text{critic}} \;+=\; c \cdot \left(Q^\pi(s, \mathrm{sg}(a_\pi)) - Q^\pi(s, a_{\text{data}}) \right). \tag{3}$$

Equation (3) penalizes the critic at the policy-generated action a_π, which is also the action whose critic gradient drives the scalar adjoint update, so the penalty controls the critic exactly where the policy update uses it. We refer to this regularizer as the *value penalty*. Unlike CQL (Kumar et al., 2020), which penalizes the critic in expectation over a sampling distribution drawn inside the critic loss, the value penalty acts at the single action the actor update has already drawn, so it adds no sampling cost.

Common forms of critic regularization act elsewhere. In-sample maximization (Kostrikov et al., 2022; Garg et al., 2023) fits the critic only on dataset actions, so it never evaluates the critic at a_π. Behavioral regularization (Tarasov et al., 2023; Fujimoto & Gu, 2021) penalizes the distance

Algorithm 1 TRQAM (Dong et al., 2026c)

Require: v^{base}, Q^{π}_{ϕ}, KL budget ε_{KL}, training steps N

$v^{\text{ft}}_{\theta} \leftarrow v^{\text{base}}$
for $n = 0, \dots, N-1$ **do**
  Sample a trajectory $X_0, \dots, X_1$ by v^{ft}_{θ}
  $X_\tau \leftarrow$ the states of that trajectory
  Solve $\mathrm{d}\tilde{a}_\tau = -\tilde{a}_\tau^{\top} \nabla_{X_\tau} \left(2v^{\text{base}} - \frac{1}{\tau} X_\tau\right) \mathrm{d}\tau$
  Update θ by Equation (2)
  TD update of Q^{π}_{ϕ}
  Dual update of λ_n for trust region
end for

Algorithm 2 SQAM (ours)

Require: v^{base}, Q^{π}_{ϕ}, KL budget ε_{KL}, training steps N

$v^{\text{ft}}_{\theta} \leftarrow v^{\text{base}}$
for $n = 0, \dots, N-1$ **do**
  Sample the endpoint X_1 by v^{ft}_{θ}
  $X_\tau \leftarrow (1-\tau)\,\epsilon + \tau X_1,\ \epsilon \sim \mathcal{N}(0, I)$
  $\hat{a}_\tau \leftarrow -\tau\, \nabla_{X_1} Q^{\pi}(s, X_1),\ \forall \tau \in [0, 1]$
  Update θ by Equation (2)
  TD update of Q^{π}_{ϕ} + Equation (3)
  Dual update of λ_n for trust region
end for

Figure 3: Simplified versions of the TRQAM and SQAM algorithms. The parts SQAM changes are marked in blue, and Section A gives the full algorithm.

between a_π and a_{data}, which constrains the actions rather than the value assigned to them. Neither controls the critic at the action whose gradient the scalar adjoint uses, and the value penalty does. Experiments in Section 4.3 support this.

4 Experiments

We evaluate SQAM on off-policy fine-tuning of pretrained flow policies in the offline and offline-to-online settings, using the 50 OGBench tasks for the main comparison. The experiments are organized around the following five questions.

- How does SQAM perform against prior methods in offline and offline-to-online RL (Section 4.1)?
- How does the scalar adjoint compare with the exact adjoint (Section 4.2)?
- Why does the value penalty work better than other value learning schemes (Section 4.3)?
- How do the value penalty coefficient c and the KL budget ε_{KL} affect SQAM (Section 4.4)?
- Does SQAM extend to large pretrained policies (Section 4.5)?

4.1 Main Experiments

Setup. OGBench (Park et al., 2025a) is a goal-conditioned RL benchmark, and we use its reward-based single-task variants over 10 domains with 5 tasks per domain, 50 tasks in all. Manipulation domains use action-chunked policies of chunk size 5 (Li et al., 2025a). All methods start from the same flow policy, pretrained with behavior cloning for 300K steps, and are fine-tuned offline for 1M steps with online runs continuing to 1.5M steps. We report the average success rate (%) over 8 seeds, and the detailed hyperparameters are in Section D.2. We compare against seven off-policy fine-tuning methods for flow policies, FQL (Park et al., 2025b), CGQL-L (Dhariwal & Nichol, 2021), DSRL (Wagenmaker et al., 2025), IFQL, QAM and QAM-E (Li & Levine, 2026), and TRQAM (Dong et al., 2026c). The last three belong to the same adjoint matching family as SQAM. See Section B for the detailed description of each baseline.

Results. Table 1 reports offline success rates at 1M steps. SQAM improves most on the four hardest domains, adding 21 points over the best baseline on `antmaze-giant`, 18 on `humanoidmaze-large`, 22 on `cube-triple`, and 35 on `cube-quadruple`, where the strongest baseline reaches 19 and SQAM reaches 54. Over all fifty tasks it reaches 75% against 68% for TRQAM, the strongest baseline. This trend holds through the online phase as well. The per-task breakdown of all fifty tasks is in Table 8, and their success rate curves are in Section G.

4.2 Comparison between Adjoints

Figure 4 explains why the scalar adjoint works even though it approximates the exact adjoint. The approximation preserves the two properties the update relies on. In magnitude, both adjoints contract in the same way as the flow time approaches zero, so the regression target in Equation (2) has a similar magnitude at each flow step under either adjoint. In direction, the scalar adjoint stays

Table 1: **Offline RL on 50 OGBench (Park et al., 2025a) tasks at 1M training steps (8 seeds).** Mean success rate (%) ± one standard deviation. Domain abbreviations are listed in Section D.1, and the per-task breakdown across all 50 tasks is in Table 8.

		al (5 tasks)	ag (5 tasks)	hm (5 tasks)	hl (5 tasks)	scene (5 tasks)	p33 (5 tasks)	p44 (5 tasks)	c2 (5 tasks)	c3 (5 tasks)	c4 (5 tasks)	all (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
	TRQAM	89 ±4	41 ±4	84 ±3	36 ±4	79 ±1	100 ±0	99 ±1	81 ±3	50 ±5	19 ±5	68
OURS	SQAM	**91** ±3	**62** ±4	**94** ±2	**54** ±5	**79** ±0	**100** ±0	85 ±6	58 ±3	**72** ±4	**54** ±5	**75**

positively aligned with the exact adjoint at every flow time. These two properties allow the scalar adjoint to stand in for the exact adjoint, but they do not make the two identical. The angle between them is larger on the two manipulation domains than on the two locomotion domains, and these are exactly the domains where the scalar adjoint alone underperforms (Section 3.2). Since the scalar adjoint relies directly on the critic's gradient, this correspondence led us to examine the role of the value learning scheme in this framework, which Section 4.3 presents.

4.3 ANALYSES OVER VALUE LEARNING SCHEMES

To justify the choice of the value penalty, we compare it with three alternative value learning schemes. Two are widely used forms of critic regularization, and the third directly regularizes the critic's action gradient that the scalar adjoint uses. The first is an in-sample critic in the style of IQL (Kostrikov et al., 2022), which fits a state value by expectile regression on dataset actions and trains the critic by TD regression toward it. The second is ReBRAC (Tarasov et al., 2023)'s bootstrap penalty, which subtracts $\|a_\pi - a_{\text{data}}\|^2$ from the bootstrap target. The third penalizes the norm of the critic's action gradient, adding $\left\|\nabla_a Q^\pi(s, \text{sg}(a_\pi))\right\|^2$ to the critic loss. See Section C for the detailed description of each scheme.

Figure 5 compares the schemes on `cube-triple` and `cube-quadruple`, the two manipulation domains where the scalar adjoint alone underperforms, and Section F.6 reports every scheme swept over its coefficient on all ten domains. The value penalty works best on both, while the in-sample critic and ReBRAC stay close to the scalar adjoint without regularization. This follows from where each scheme acts. The scalar adjoint uses the critic's gradient at the policy-generated action a_π directly at every flow step, so the policy update depends directly on the critic at a_π. On the other hand, the in-sample critic is trained only on dataset actions, so it never evaluates the critic at a_π. ReBRAC constrains the distance between a_π and a_{data}, which controls the action itself rather than the value the critic assigns to it. Only the value penalty acts on the critic's value at a_π.

The gradient-norm penalty also improves over the unregularized critic, most clearly on `cube-quadruple`, which is consistent with its close relation to the value penalty. A first-order expansion of the value penalty in Equation (3) around a_π gives

$$Q^\pi(s, a_\pi) - Q^\pi(s, a_{\text{data}}) \approx \nabla_{a_\pi} Q^\pi(s, a_\pi)^\top (a_\pi - a_{\text{data}}),$$

so the two penalties are linked through the critic's gradient at a_π.

The explanation above rests on the scalar adjoint using the critic's gradient at a_π directly. The exact adjoint instead passes this gradient through the velocity Jacobian at every flow step, so if the explanation holds, the value penalty should help the exact adjoint less. To test this, we also apply the value penalty to TRQAM, which uses the exact adjoint. The value penalty improves TRQAM as well, but far less than it improves SQAM (Section F.2).

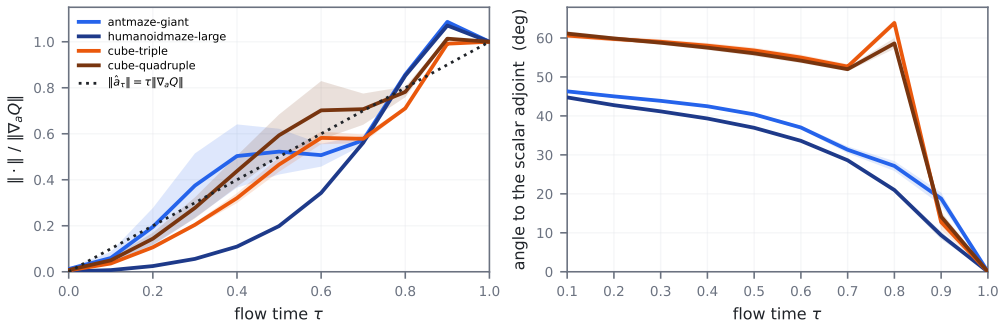


Figure 4: Exact and scalar adjoints computed on the same rollout states. **Left:** the norm of each adjoint relative to $\|\nabla_a Q\|$, where the scalar adjoint is the dotted line τ. **Right:** the angle between the two adjoints. The two adjoints contract similarly in norm and stay positively aligned. The angle is larger on the manipulation domains, where the scalar adjoint alone underperforms. Three seeds per domain at 0.3M steps, without the value penalty.

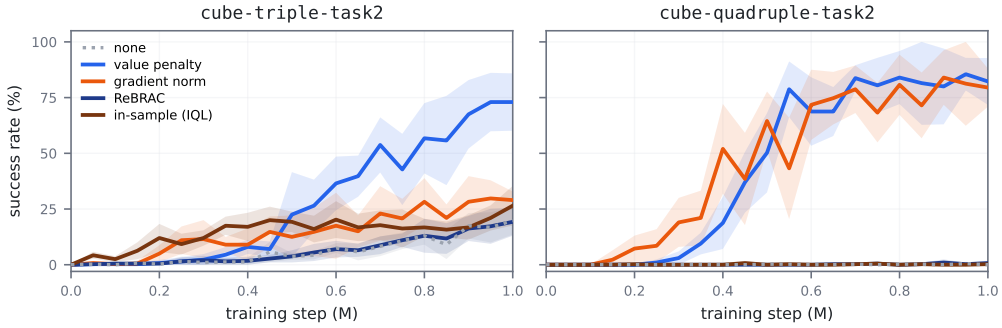


Figure 5: Value learning schemes compared on the two domains where the scalar adjoint alone fails. Each scheme is shown at its best coefficient on the grid we swept, and the full sweeps on all ten domains are in Section F.6. The value penalty lifts both domains and the gradient-norm penalty follows it on `cube-quadruple`, while ReBRAC and the in-sample critic stay close to the scalar adjoint without regularization. Eight seeds per curve, and bands are one standard deviation.

4.4 Ablation Studies

Effect of value penalty c. Figure 6 assesses the value penalty coefficient c on one task from each of four domains. On the two cube manipulation tasks, the success rate rises sharply as c grows, while on the two locomotion tasks the gains are small. These results indicate that the value penalty matters mainly on the domains where the scalar adjoint alone underperforms. We use $c = 0.3$ on every domain except `cube-double`, where the penalty is off because a large c lowers its offline performance (Section F.5).

Effect of KL budget $\epsilon_{\mathbf{KL}}$. The budget directly sets how far the fine-tuned policy may deviate from the pretrained one (Dong et al., 2026c). We sweep it on all ten domains (Section F.5), and Figure 7 shows four of them. On most domains, performance stays flat or decreases steadily as the budget widens, so the budget affects performance predictably and a small budget is a reliable default. The exception is `puzzle-4x4`, which prefers a larger budget, consistent with its larger state space.

4.5 Extension to Large Pretrained Policies

To test whether SQAM extends to large pretrained policies, we fine-tune RLDX-1 (Kim et al., 2026b), a pretrained vision-language-action policy, on a real-world bimanual robot (Rainbow Robotics, 2024) equipped with two 20-DoF dexterous hands (Wuji Technology, 2025), with an action chunk of 16 steps. Figure 8 shows the three manipulation tasks. Starting from the pretrained policy, we run supervised fine-tuning on task demonstrations, then SQAM for 15K offline steps and a further 10K online steps. We compare against EXPO-FT (Dong et al., 2026a), a residual fine-

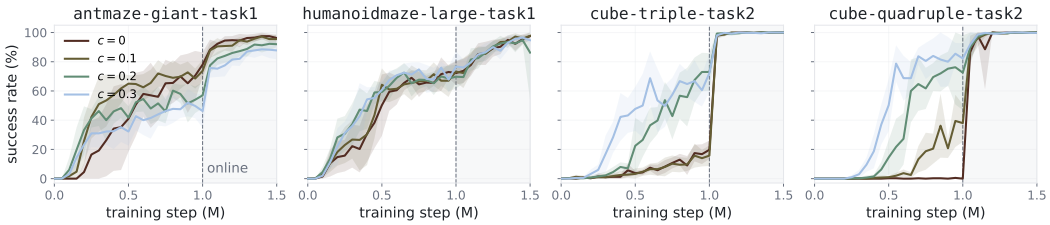


Figure 6: Sweep over value penalty c on four of the ten swept domains. Eight seeds per curve, and bands are one standard deviation. The full analysis on all ten domains is in Section F.5.

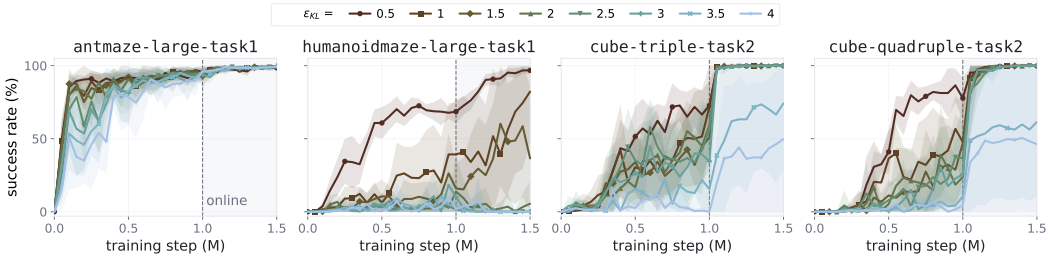


Figure 7: Sweep over KL budget ϵ_{KL} on four of the ten swept domains. Eight seeds per curve, and bands are one standard deviation. The full analysis on all ten domains is in Section F.5.

tuning method that freezes the pretrained policy and learns an edit policy that adds corrections to its actions (Section B), under the shared protocol of Section E. Table 2 reports held-out success over 30 episodes per task, and Figure 10 in Section E.4 tracks the training rollouts through the online phase.

SQAM improves over SFT on all three tasks, whereas EXPO-FT does not appear to work well in this setting. Whereas SQAM updates the flow policy directly, EXPO-FT trains a residual edit policy, which must be learned from scratch and cannot directly exploit the expressiveness of the pretrained flow policy. We conjecture that these two differences account for this.

5 Related Work

Propagating value to intermediate flow steps. A flow policy generates its action over many flow steps, while the critic evaluates only the final action. Using the critic to steer the flow therefore requires propagating its value to each intermediate step, and existing methods differ mainly in how they do so (Lee et al., 2026). The most common approach evaluates the critic at a Tweedie posterior estimate of the final action, and it is used for guidance at test time (Chung et al., 2023; Kim et al., 2025; Zhou et al., 2026). Another line avoids the approximation and feeds the intermediate action to the critic directly, either by matching the policy's score to the critic's action gradient at the intermediate action (Psenka et al., 2024) or by learning a separate value at each noise level (Oberai et al., 2026; Doo et al., 2026). Adjoint matching (Domingo-Enrich et al., 2025; Li & Levine, 2026; Dong et al., 2026c; Bergmeister et al., 2026) gives a theoretically grounded answer, propagating the value gradient back to each flow step through the lean adjoint ODE. For test-time guidance rather than fine-tuning, dropping the Jacobian has also been found to outperform backpropagating through the exact Jacobian (Zhou et al., 2026).

Critic regularization. Many value learning schemes regularize the critic, and they differ in where the regularization acts. CQL penalizes the critic's value at the actions the policy prefers relative to the data (Kumar et al., 2020), IQL fits a state value by expectile regression and trains the critic by TD regression toward it, using only dataset actions (Kostrikov et al., 2022), and ReBRAC subtracts an action-distance penalty from the bootstrap target (Tarasov et al., 2023). Fisher-BRC penalizes the action gradient of the critic's offset from a behavior model (Kostrikov et al., 2021), which is close to the gradient-norm penalty we compare in Section 4.3, and Cal-QL calibrates the critic against a reference policy for offline-to-online fine-tuning (Nakamoto et al., 2023).

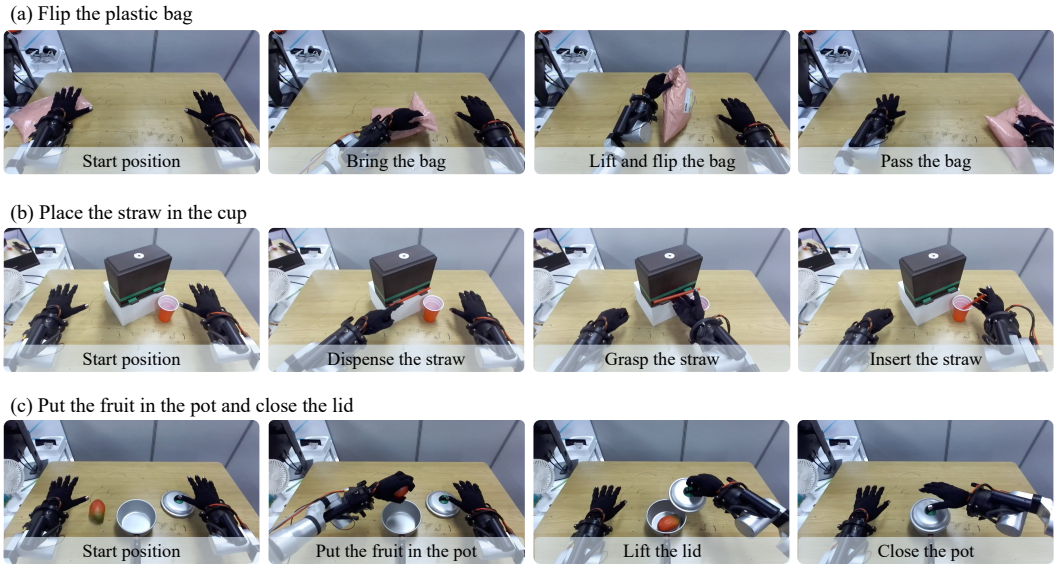


Figure 8: Overview of the three real-world manipulation tasks. From top to bottom: flipping a plastic bag, placing a straw in a cup, and placing fruit in a pot and closing the lid. Each row illustrates the task progression from left to right.

Table 2: Success rate on three real-robot manipulation tasks with a vision-language-action policy. The offline rows evaluate the checkpoint after 15K offline steps, and the offline-to-online rows the checkpoint after a further 10K online steps. Each cell counts successes over 30 held-out episodes.

Method	Flip plastic bag	Place the straw	Put the fruit and close the lid
SFT	15/30	7/30	4/30
EXPO-FT (offline)	8/30	0/30	5/30
EXPO-FT (offline2online)	15/30	0/30	3/30
SQAM (offline)	20/30	13/30	8/30
SQAM (offline2online)	**25/30**	**15/30**	**13/30**

Offline-to-online RL. Pretraining a policy and a critic on offline data and then fine-tuning them online is a standard recipe for sample-efficient RL (Ball et al., 2023; Hester et al., 2018; Hu et al., 2024; Kostrikov et al., 2022; Kumar et al., 2020; Lei et al., 2024; Luo et al., 2024; Nair et al., 2021; Nakamoto et al., 2023; Rajeswaran et al., 2018; Vecerik et al., 2018). A central challenge is the distribution shift at the transition, which destabilizes the value function and can make the policy forget what it learned offline (Ball et al., 2023; Nakamoto et al., 2023; Wołczyk et al., 2024).

6 Conclusion

We introduced SQAM, which fine-tunes flow policies with off-policy RL without a vector–Jacobian product through the policy. Motivated by the diagonal structure of the batch-averaged velocity Jacobian, it replaces the exact adjoint with a closed-form scalar adjoint and pairs it with a value penalty at the policy-generated actions. SQAM improves most on the four hardest OGBench domains and extends to a vision-language-action policy on a real robot.

Our results also suggest a broader point. A rough approximation of the adjoint suffices, while the value learning scheme largely determines how well it performs. The bottleneck of off-policy RL with flow policies may therefore lie less in the policy update than in the critic. We hope this encourages further work on value learning for flow policies.

Reproducibility statement

Section A gives the complete pseudocode of SQAM, and Section 3.1 states and proves the closed form it builds on. Section D lists the domains, datasets, and hyperparameters, and Section B describes every baseline. The OGBench experiments use the public benchmark (Park et al., 2025a) and the datasets described in Section D. Section E documents the platform, observation and action spaces, training details, and evaluation protocol of the real-robot experiments. We will release the implementation of SQAM.

Ethics statement

This work fine-tunes robot policies with reinforcement learning in simulation and on a robot in a controlled laboratory setting. The real-robot experiments involve no human subjects and no personal data, and an operator supervised the robot throughout. As with any method that improves a policy against a learned value function, deployment beyond a supervised setting calls for the usual safeguards.

AI use statement

We used generative AI tools to assist with writing, editing, and routine coding and plotting. We did not use them to generate research ideas, experimental results, or claims. The authors reviewed all AI-assisted text and code and take full responsibility for the content of this paper.

References

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

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

Andreas Bergmeister, Stefanie Jegelka, Nikolas Nüsken, Carles Domingo-Enrich, and Jakiw Pidstrigach. Reinforce adjoint matching: Scaling rl post-training of diffusion and flow-matching models. In *Advances in Neural Information Processing Systems*, 2026.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "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.

Yuhui Chen, Shuai Tian, Shugao Liu, Yingting Zhou, Haoran Li, and Dongbin Zhao. Conrft: A reinforced fine-tuning method for vla models via consistency policy. In *Robotics: Science and Systems*, 2025.

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.

Hyungjin Chung, Jeongsol Kim, Michael T. Mccann, Marc L. Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. In *International Conference on Learning Representations*, 2023.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis. In *Advances in Neural Information Processing Systems*, 2021.

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

Perry Dong, Kuo-Han Hung, Tian Gao, Dorsa Sadigh, and Chelsea Finn. Expo-ft: Sample-efficient reinforcement learning finetuning for vision-language-action models. *arXiv preprint arXiv:2605.25477*, 2026a.

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

Yonghoon Dong, Kyungmin Lee, Changyeon Kim, Jaehyuk Kim, and Jinwoo Shin. Trust region q adjoint matching. In *Advances in Neural Information Processing Systems*, 2026c.

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.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Divyansh Garg, Joey Hejna, Matthieu Geist, and Stefano Ermon. Extreme q-learning: Maxent rl without entropy. In *International Conference on Learning Representations*, 2023.

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.

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.

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

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*, 2025a.

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*, 2025b.

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*, 2026a.

Dongyoung Kim, Huiwon Jang, Myungkyu Koo, Suhyeok Jang, Taeyoung Kim, Beomjun Kim, Byungjun Yoon, Changsung Jang, Daewon Choi, Dongsu Han, Donguk Lee, Heeseung Kwon, Hojin Jeon, Jaehyun Kang, Jaekyoung Bae, Jihyuk Lee, Jimin Lee, John Won, Joonwoo Ahn, Junhyeong Park, Junyoung Sung, Kyungmin Lee, Minseong Han, Minsung Yoon, Sejune Joo, Seonil Son, Seungcheol Park, Seunggeun Cho, Seungjun Moon, Seungku Kim, Yonghoon Dong,

Yongjin Cho, Youngchan Kim, Chang Hwan Kim, Dohyeon Kim, Heecheol Kim, Heewon Lee, Hensen Ahn, Hyungkyu Ryu, Hyunsoo Choi, Hyunsoo Shin, Jaeheon Jung, Jaewoo Kim, Jinwook Kim, Joochul Chang, Joonsoo Kim, Junghun Park, Jungwoo Park, Junho Cho, Junhyeok Park, Junwon Lee, Kangwook Lee, Kwanghoon Kim, Kyoungwhan Choe, Manoj Bhadu, Nayoung Oh, Sangjun Kim, Sangwoo Kim, Seunghoon Shim, Seunghyun Kim, Seungjun Lee, Seungyup Ka, Sungryol Yang, Wook Jung, Yashu Shukla, Yeonjae Lee, Yeonwoo Bae, and Jinwoo Shin. Rldx-1 technical report. *arXiv preprint arXiv:2605.03269*, 2026b.

Jeongsol Kim, Bryan Sangwoo Kim, and Jong Chul Ye. Flowdps: Flow-driven posterior sampling for inverse problems. In *IEEE International Conference on Computer Vision*, 2025.

Ilya Kostrikov, Jonathan Tompson, Rob Fergus, and Ofir Nachum. Offline reinforcement learning with fisher divergence critic regularization. In *International Conference on Machine Learning*, 2021.

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

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

Jeongjae Lee, Jinho Chang, Jeongsol Kim, and Jong Chul Ye. Reward score matching: Unifying reward-based fine-tuning for flow and diffusion models. In *Advances in Neural Information Processing Systems*, 2026.

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.

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

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

Yunfei Li, Xiao Ma, Jiafeng Xu, Yu Cui, Zhongren Cui, Zhigang Han, Liqun Huang, Tao Kong, Yuxiao Liu, Hao Niu, Wanli Peng, Jingchao Qiao, Zeyu Ren, Haixin Shi, Zhi Su, Jiawen Tian, Yuyang Xiao, Shenyu Zhang, Liwei Zheng, Hang Li, and Yonghui Wu. Gr-rl: Going dexterous and precise for long-horizon robotic manipulation. *arXiv preprint arXiv:2512.01801*, 2025b.

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

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.

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.

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.

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

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.

Aditya Oberai, Seohong Park, and Sergey Levine. Reversal q-learning. *arXiv preprint arXiv:2606.17551*, 2026.

Seohong Park, Kevin Frans, Benjamin Eysenbach, and Sergey Levine. Ogbench: Benchmarking offline goal-conditioned rl. In *International Conference on Learning Representations*, 2025a.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *International Conference on Machine Learning*, 2025b.

Ahmad Parsian and SNUA Kirmani. Estimation under linex loss function. In *Handbook of applied econometrics and statistical inference*, pp. 75–98. CRC Press, 2002.

Xinyu Peng, Ziyang Zheng, Wenrui Dai, Nuoqian Xiao, Chenglin Li, Junni Zou, and Hongkai Xiong. Improving diffusion models for inverse problems using optimal posterior covariance. In *International Conference on Machine Learning*, 2024.

Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. In *International Conference on Machine Learning*, 2024.

Rainbow Robotics. RB-Y1: Dual-arm mobile manipulator. https://rainbow-robotics.com/en/products/rb-y1/, 2024.

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.

Stereolabs. ZED Mini: Stereo camera. https://www.stereolabs.com/store/products/zed-mini, 2018.

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

Denis Tarasov, Vladislav Kurenkov, Alexander Nikulin, and Sergey Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, 2023.

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.

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.

Yi Wang, Xinchen Li, Pengwei Xie, Pu Yang, Buqing Nie, Yunuo Cai, Qinglin Zhang, Chendi Qu, Jeffrey Wu, Jianheng Song, Xinlin Ren, Jingshun Huang, Mingjie Pan, Siyuan Feng, Zhi Chen, and Jianlan Luo. Learning while deploying: Fleet-scale reinforcement learning for generalist robot policies. In *RSS Workshop on Reinforcement Learning for Vision-Language-Action Models (RL4VLA)*, 2026.

Maciej Wołczyk, Bartłomiej Cupiał, Mateusz Ostaszewski, Michał Bortkiewicz, Michał Zajac, 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.

Wuji Technology. Wuji Hand 2: 20-DoF dexterous robotic hand. https://www.wuji.tech/en/hand2, 2025.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi "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.

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.

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.

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. In *Advances in Neural Information Processing Systems*, 2026.

CONTENTS

A Full algorithm **16**

B Baselines **17**

C Value learning schemes **18**

D Experimental details **19**

D.1 Domains and tasks 19

D.2 Hyperparameters 20

E Real-robot experiments: RB-Y1 with Wuji Hand 2 **20**

E.1 Platform, observations and actions 21

E.2 Tasks and evaluation protocol 22

E.3 Training details 22

E.4 Additional results 22

F Additional experiments **26**

F.1 The cost of one update 26

F.2 The value penalty inside TRQAM 26

F.3 Without the trust region 26

F.4 Batch-averaged velocity Jacobian on all domains 27

F.5 Ablation studies on all domains 27

F.6 Value learning schemes on all domains 27

G Full OGBench results **27**

H Limitations and future work **39**

A FULL ALGORITHM

Algorithm 3 presents the complete pseudocode of SQAM, including the trust region it inherits from TRQAM (Dong et al., 2026c). Blue marks what SQAM changes from TRQAM.

The scalar adjoint also changes which states the policy loss is evaluated on. The exact adjoint integrates the lean adjoint ODE in Equation (1) backward along a sampled trajectory,

$$\mathrm{d}\tilde{a}_\tau = -\tilde{a}_\tau^\top \nabla_{X_\tau}\left(2v^{\mathrm{base}}(X_\tau,\tau) - \frac{1}{\tau}X_\tau\right)\mathrm{d}\tau, \qquad \tilde{a}_1 = -\nabla_{X_1} Q^\pi(s, X_1),$$

so the adjoint at each flow time depends on the later ones, and the loss states must be the states of that trajectory. The scalar adjoint removes this coupling, so the loss states can be built from the final action alone. Following Bergmeister et al. (2026), we sample the final action with the ODE sampler instead of a long and numerically unstable stochastic rollout, and re-noise it along the linear interpolation path, $X_\tau = (1-\tau)\,\epsilon + \tau\, a_\pi$ with fresh $\epsilon \sim \mathcal{N}(0, I)$. We call these the bridge states, which match the rollout-state distribution at the optimum of the stochastic optimal control problem and approximate it elsewhere (Bergmeister et al., 2026). In addition, the trust region is optional, and we use it by default for stable training (Section F.3).

Algorithm 3 Q-learning with Scalar Adjoint Matching (SQAM)

Input: replay buffer $\mathcal{D}$; v^{base}: pretrained velocity field; v_θ^{ft}: fine-tuned velocity field; Q_ϕ: critic function; step size h; KL budget $\varepsilon_{\mathrm{KL}}$; dual step size η_λ; EMA coefficient ρ; value penalty coefficient c; fine-tuning iterations N.
Initialize: $v_\theta^{\mathrm{ft}} \leftarrow v^{\mathrm{base}}$ with parameters θ; $\lambda_0 > 0$; $\overline{D}_0 \leftarrow 0$.
Noise schedule: $\sigma_n(\tau) := g(\tau)/\sqrt{\lambda_n}$, where $g(\tau) := \sqrt{2(1-\tau)/\tau}$
for $n \in \{0, \dots, N-1\}$ **do**
  Sample a batch $\mathcal{B} = \{(s_i, a_i, r_i, s_i')\}$ from $\mathcal{D}$
  Rollout: For each state $s \in \mathcal{B}$, sample the endpoint X_1 by the flow itself:

$$X_{\tau+h} = X_\tau + h\, v_\theta^{\mathrm{ft}}(s, X_\tau, \tau), \quad X_0 \sim \mathcal{N}(0, I),$$

and set $a_\pi \leftarrow \texttt{stopgrad}(X_1)$.
  Bridge states: $X_\tau \leftarrow (1-\tau)\,\epsilon + \tau\, a_\pi$ with fresh $\epsilon \sim \mathcal{N}(0, I)$, for all $\tau \in \{h, 2h, \dots, 1\}$
  Scalar adjoint: $\hat{a}_\tau \leftarrow -\tau\, \nabla_a Q_\phi(s, a_\pi)$ for all $\tau \in \{h, 2h, \dots, 1\}$
  Stop gradient: $X_\tau \leftarrow \texttt{stopgrad}(X_\tau)$, $\hat{a}_\tau \leftarrow \texttt{stopgrad}(\hat{a}_\tau)$.
  Critic update: Optimize ϕ w.r.t.

$$\mathcal{L}(\phi) = \frac{1}{|\mathcal{B}|}\sum_{(s,a,r,s')\in\mathcal{B}}\Big[\big(Q_\phi(s,a) - r - \gamma\, Q_{\bar{\phi}}(s',\, a' \sim \pi_\theta(\cdot \mid s'))\big)^2 + c\,\big(Q_\phi(s, a_\pi) - Q_\phi(s,a)\big)\Big]$$

  Policy update: Optimize θ w.r.t. the adjoint matching objective:

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

  Trust region update:
  Estimate path-space KL surrogate:

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

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

B BASELINES

On OGBench, we compare against seven off-policy fine-tuning methods for flow policies, spanning the main fine-tuning paradigms. On the real robot, we compare against EXPO-FT (Section 4.5). All of them read transitions (s, a, r, s') uniformly from a replay buffer $\mathcal{D}$, which holds the offline dataset during offline training and grows with the collected rollouts online.

FQL (Park et al., 2025b). FQL avoids differentiating through the sampling chain by distilling the multi-step flow into a one-step policy. The behavior-cloning rollout follows the ODE

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

and we denote its terminal value by $\mathrm{ODE}(v_\theta, s, X_0) := X_1$. A one-step policy $\pi_\omega(s, X_0)$ trains jointly with v_θ, maximizing the critic while an ℓ_2 term holds it near the rollout,

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

The coefficient α sets how closely π_ω follows the distillation target, and π_ω is the policy that acts in the environment.

CGQL-L (Dhariwal & Nichol, 2021). CGQL-Linex is the guidance baseline introduced in Li & Levine (2026). It combines the BC velocity field with classifier-free guidance (Dhariwal & Nichol, 2021) from a Q-function. An auxiliary critic $Q_\psi(s, X_\tau, \tau)$ is trained on intermediate noisy actions and induces the guidance velocity

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

where $X_\tau = (1-\tau)X_0 + \tau X_1$ with $X_0 \sim \mathcal{N}(0, I)$. Sampling runs the combined velocity $v = v^{\mathrm{base}} + w\, \hat{v}_\psi$ with guidance weight w, steering toward the entropy-regularized target $\pi^\star(\cdot \mid s) \propto e^{\beta Q_\phi(s,\cdot)}$. The auxiliary critic trains with a Linex regression (Parsian & Kirmani, 2002; Myers et al., 2025),

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

and the standard critic Q_ϕ trains by TD with target actions sampled from the combined velocity,

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

Following Li & Levine (2026), the Linex loss carries a Huber-style stabilization against exponential blow-up.

DSRL (Wagenmaker et al., 2025). DSRL leaves the flow frozen and runs RL in its noise space. A one-step Gaussian noise-space policy $\pi_\omega(X_0 \mid s)$ trains with SAC against a noise-space critic $Q_\psi(s, X_0)$ that regresses to the action-space critic,

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

At inference the noise is drawn from the noise-space policy, $X_0 \sim \pi_\omega(\cdot \mid s)$, and pushed through the flow, $a = \mathrm{ODE}(v_\theta, s, X_0)$. Following Li & Levine (2026), our DSRL also fine-tunes the BC velocity online, with a target network $v_{\bar{\theta}}$ for stability, which strengthens its offline-to-online performance.

IFQL. IFQL is the flow counterpart of implicit diffusion Q-learning (Hansen-Estruch et al., 2023), taken as a baseline in Park et al. (2025b). Value learning is IQL-style expectile regression (Kostrikov et al., 2022). A value network V_ξ fits an upper expectile of the critic, and the critic bootstraps through it,

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

where $L_2^\kappa(u) = |\kappa - \mathbf{1}(u < 0)|\, u^2$ is the expectile loss with $\kappa \in (0.5, 1)$. The policy draws N candidate actions from the BC flow, $a^{(i)} = \mathrm{ODE}(v_\theta, s, X_0^{(i)})$, and acts with the one the critic values highest.

QAM and QAM-E (Li & Levine, 2026). Adjoint matching casts fine-tuning as stochastic optimal control, which adds a control u to the drift b of the sampler,

$$\mathrm{d}X^u_\tau = \big(b(X^u_\tau,\tau) + \sigma(\tau)u(X^u_\tau,\tau)\big)\,\mathrm{d}\tau + \sqrt{\lambda}\,\sigma(\tau)\,\mathrm{d}B_\tau, \tag{4}$$

with B_τ a Brownian motion and λ a scale on the diffusion, and the base sampler is Equation (4) at $u = 0$. QAM solves this control problem at a constant λ, with an inverse temperature β on the terminal reward $\beta\, Q_\phi(s, X_1)$. The fine-tuned velocity trains against the base velocity through the lean adjoint matching loss

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

where $\tilde{a}_\tau$ is the lean adjoint with terminal condition $\tilde{a}_1 = -\beta\,\nabla_{X_1} Q_\phi(s, X_1)$ and X_τ follows the sampling dynamics of Equation (4) at that constant λ. Element-wise gradient clipping keeps the update numerically stable. QAM-E adds a residual edit policy $\pi_\omega(\Delta a \mid s, \tilde{a})$ on top, perturbing the QAM-generated action $\tilde{a}$ by at most σ_a in L_∞ distance through a tanh-squashed Gaussian, trained with entropy-regularized SAC and automatic entropy tuning.

TRQAM (Dong et al., 2026c). TRQAM is the trust-region member of the same family and the strongest baseline. Where QAM holds the temperature fixed, TRQAM reads λ in Equation (4) as a trust-region parameter on the sampling dynamics and adapts it over training. Girsanov's theorem gives the path-space KL between the controlled and the base dynamics in closed form,

$$D_{\mathrm{KL}}\big(\mathbb{P}^u \,\|\, \mathbb{P}^{\mathrm{base}}\big) \;=\; \mathbb{E}_{\mathbf{X}\sim\mathbb{P}^u}\left[\int_0^1 \frac{2}{\lambda\,\sigma(\tau)^2}\,\big\|v^{\mathrm{ft}}_\theta(X^u_\tau,\tau) - v^{\mathrm{base}}(X^u_\tau,\tau)\big\|^2\,\mathrm{d}\tau\right], \tag{5}$$

so the deviation can be measured on every rollout at no additional cost. Each update estimates the path-space KL of Equation (5) on its own rollout, smooths the estimate with an EMA $\overline{D}_n$, and takes the projected dual step

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

with step size η_λ. The constraint therefore lives inside the sampling dynamics, held at the prescribed budget $\varepsilon_{\mathrm{KL}}$, rather than as a loss-level penalty that a strong critic signal can override.

EXPO and EXPO-FT (Dong et al., 2026b;a). EXPO trains the expressive base policy π_{base} with a stable imitation objective and never maximizes the value through it. A light-weight Gaussian edit policy $\pi_\omega(\Delta \mid s, a)$ instead adds a residual to the base actions, $\tilde{a} = a + \beta\Delta$ with edit scale β, and is trained with an entropy-regularized objective to raise $Q_\phi(s, \tilde{a})$. Actions are chosen by an on-the-fly policy that draws N base actions $a_{1:N} \sim \pi_{\mathrm{base}}(\cdot \mid s)$, edits each of them, and takes the candidate with the highest value,

$$a^\star = \operatorname*{arg\,max}_{a\in\{a_i,\tilde{a}_i\}_{i=1}^N} Q_\phi(s, a),$$

both for acting and for the bootstrap action in the TD target. EXPO-FT applies this recipe to a pretrained vision-language-action policy (Dong et al., 2026a). We use EXPO-FT as the baseline of Section 4.5, with the settings of Section E.3.

C VALUE LEARNING SCHEMES

All schemes compared in Figure 5 run inside the same SQAM implementation and differ only in how the critic is trained. The default critic loss is the temporal-difference loss

$$\mathcal{L}_{\mathrm{TD}}(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\Big[\big(Q^\pi_\phi(s,a) - r - \gamma\, Q^\pi_{\bar{\phi}}(s', a')\big)^2\Big], \qquad a' \sim \pi_\theta(\cdot \mid s'), \tag{6}$$

where a_π denotes the final action the actor update has already drawn from $\pi_\theta(\cdot \mid s)$. The value penalty and the gradient-norm penalty add a term to Equation (6), ReBRAC modifies its bootstrap target, and the in-sample critic replaces it with a value target fit by expectile regression. We summarize each scheme below.

Value penalty. Adds to Equation (6) the gap between the critic's value at the policy's own action and at the dataset action, as in Equation (3),

$$\mathcal{L}(\phi) = \mathcal{L}_{\mathrm{TD}}(\phi) + c\,\mathbb{E}_{s\sim\mathcal{D}}\left[Q^{\pi}_{\phi}(s, \mathrm{sg}(a_{\pi})) - Q^{\pi}_{\phi}(s, a_{\mathrm{data}})\right].$$

The stop-gradient keeps the penalty's gradient off the policy, and reusing a_π means the penalty draws no additional sample (Section 3.2).

Gradient-norm penalty. Constrains the magnitude of the gradient the scalar adjoint uses rather than the value penalty,

$$\mathcal{L}(\phi) = \mathcal{L}_{\mathrm{TD}}(\phi) + c_g\,\mathbb{E}_{s\sim\mathcal{D}}\left[\left\|\nabla_a Q^{\pi}_{\phi}(s, \mathrm{sg}(a_{\pi}))\right\|^2\right].$$

A first-order expansion of the value penalty relates the two penalties (Section 4.3).

In-sample critic (Kostrikov et al., 2022). Replaces the bootstrap action with a state value fit by expectile regression, so the critic is never evaluated off the dataset's actions,

$$\mathcal{L}_V(\xi) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[L_2^{\kappa}\left(Q^{\pi}_{\bar{\phi}}(s,a) - V_{\xi}(s)\right)\right],$$
$$\mathcal{L}(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q^{\pi}_{\phi}(s,a) - r - \gamma\, V_{\xi}(s')\right)^2\right],$$

with $L_2^{\kappa}(u) = |\kappa - \mathbf{1}(u<0)|\,u^2$.

ReBRAC's bootstrap penalty (Tarasov et al., 2023). Subtracts the distance between the policy's next-state action and the dataset's from the bootstrap target, so the control acts on the actions rather than on the value the critic assigns them,

$$\mathcal{L}(\phi) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q^{\pi}_{\phi}(s,a) - r - \gamma\left(Q^{\pi}_{\bar{\phi}}(s', a') - \beta\|a' - a'_{\mathrm{data}}\|^2\right)\right)^2\right],$$

with $a' \sim \pi_\theta(\cdot \mid s')$ and a'_{data} the dataset action at s'.

D Experimental Details

D.1 Domains and Tasks

We evaluate on OGBench (Park et al., 2025a), an offline goal-conditioned RL benchmark, using its reward-based single-task variants. We use all 10 domains, spanning long-horizon navigation, multi-object manipulation, and combinatorial planning, with 5 tasks per domain and 50 tasks in all. The domains are abbreviated as `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`).

The dataset size, episode length, and action dimension of 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 $\pm$ one standard deviation across seeds, and plots show the mean with shaded regions denoting the same.

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

Dataset sources. We use the official OGBench datasets (Park et al., 2025a) except where noted, and follow the dataset choices of Dong et al. (2026c) for the three larger ones. For `cube-triple-10M-*` and `puzzle-4x4-10M-*` we use a 10M subset of the official 100M release, which is split into 100 files of 1M transitions each, taking the first 10 files sorted by name (Kim et al., 2026a; Dong et al., 2026c). For `antmaze-giant-10M-*`, OGBench does not release a dataset at this size, so we use the dataset of Dong et al. (2026c), which was generated with the official OGBench data-generation pipeline at default settings.

D.2 HYPERPARAMETERS

Most methods share the common hyperparameters in Table 4, and the domain-specific ones are listed in Table 5. The baseline results are drawn from the exported runs of Dong et al. (2026c), which adopt the per-domain values reported as optimal by Li & Levine (2026) for FQL, DSRL, IFQL, CGQL-L, QAM, and QAM-E, and tune $\varepsilon_{\mathrm{KL}}$ for TRQAM. SQAM adopts the per-domain KL budgets of TRQAM, including the time-varying schedule on `antmaze-giant`, $\varepsilon_{\mathrm{KL}} = 0.5$ offline opening to 3.0 at the online transition. We tune the value penalty coefficient following the protocol of Dong et al. (2026c), comparing $c \in \{0, 0.3\}$ on two tuning tasks per domain, task 1 and task 4 for locomotion and task 2 and task 4 for manipulation, with 2 seeds per configuration that differ from the evaluation seeds. On the two domains where TRQAM uses $\varepsilon_{\mathrm{KL}} = 1.0$, `cube-quadruple` and `antmaze-large`, we additionally check $\varepsilon_{\mathrm{KL}} = 0.5$ under the same protocol, and adopt it on `cube-quadruple`. The selected values are listed in Table 5, and all reported results average over 8 evaluation seeds.

Table 3: **Domain metadata.** Following OGBench, we group the domains into locomotion and manipulation. Action dimensions are per control step, and the manipulation domains use action chunks of size 5.

Domain	Category	Data	Horizon	Act. dim.
`cube-double-*`	Manipulation	1M	500	5
`cube-triple-10M-*`	Manipulation	10M	1000	5
`cube-quadruple-100M-*`	Manipulation	100M	1000	5
`antmaze-large-*`	Locomotion	1M	1000	8
`antmaze-giant-10M-*`	Locomotion	10M	1000	8
`humanoidmaze-medium-*`	Locomotion	1M	2000	21
`humanoidmaze-large-*`	Locomotion	1M	2000	21
`scene-*`	Manipulation	1M	750	5
`puzzle-3x3-*`	Manipulation	1M	500	5
`puzzle-4x4-10M-*`	Manipulation	10M	500	5

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
Gradient max-norm clipping	False (default), 1 (`QAM`, `QAM-E`, `TRQAM`, `SQAM`)
Actor layer norm	False (default), True (10M/100M data)
Critic layer norm	True

E REAL-ROBOT EXPERIMENTS: RB-Y1 WITH WUJI HAND 2

This appendix documents the real-robot experiments on a dual-arm RB-Y1 humanoid (Rainbow Robotics, 2024) equipped with two Wuji Hand 2 dexterous hands (Wuji Technology, 2025). For each of three manipulation tasks we follow the same pipeline: (i) behavior cloning (BC) of a vision-language-action base policy on teleoperated demonstrations, (ii) collection of 30 rollout episodes of the BC policy on the robot, (iii) offline RL for 15K steps on the demonstrations plus these 30 BC

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 $\varepsilon_{\mathrm{KL}}$	SQAM $(\varepsilon_{\mathrm{KL}}, c)$
`scene-*`	300	0.4	0.9	$(10, 0.1, 0.1)$	1	$(1, 0)$	0.5	$(0.5, 0.3)$
`puzzle-3x3-*`	300	1.0	0.95	$(10, 0.001, 0.1)$	3	$(1, 0.1)$	2.0	$(2.0, 0.3)$
`puzzle-4x4-10M-*`	1	1.0	0.9	$(10, 0.001, 1)$	30	$(0.1, 0.9)$	4.0	$(4.0, 0.3)$
`cube-double-*`	300	1.0	0.9	$(10, 0.001, 0.01)$	1	$(1, 0)$	0.5	$(0.5, 0)$
`cube-triple-10M-*`	30	1.4	0.95	$(10, 0.001, 0.1)$	3	$(3, 0.1)$	0.5	$(0.5, 0.3)$
`cube-quadruple-100M-*`	100	1.4	0.95	$(10, 0.1, 0.01)$	1	$(3, 0.1)$	1.0	$(0.5, 0.3)$
`antmaze-large-*`	3	0.8	0.9	$(10, 0.001, 0.1)$	10	$(1, 0.1)$	1.0	$(1.0, 0.3)$
`antmaze-giant-10M-*`	3	1.2	0.8	$(10, 0.001, 0.1)$	3	$(10, 0.1)$	$0.5 \to 3.0$	$(0.5 \to 3.0, 0.3)$
`humanoidmaze-medium-*`	30	0.6	0.7	$(10, 0.1, 0.1)$	3	$(3, 0.1)$	0.5	$(0.5, 0.3)$
`humanoidmaze-large-*`	30	0.8	0.8	$(10, 0.1, 0.1)$	3	$(3, 0.1)$	0.5	$(0.5, 0.3)$

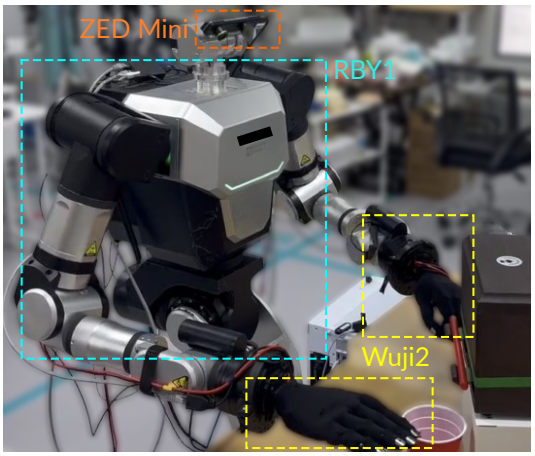


Figure 9: Hardware setup: RB-Y1 humanoid with a Wuji Hand 2 on each wrist and a head-mounted ZED Mini stereo camera.

rollout episodes, (iv) online RL for a further 10K steps that continues from the offline checkpoint while the robot collects new episodes, and (v) a 30-episode evaluation of every checkpoint. The same protocol is run for our method (SQAM) and for the EXPO-FT baseline (Dong et al., 2026a).

E.1 Platform, observations and actions

Hardware. We use a Rainbow Robotics RB-Y1 (Rainbow Robotics, 2024) dual-arm humanoid with a 20-DoF Wuji Hand 2 (Wuji Technology, 2025) mounted on each wrist. The experiments use 54 of its degrees of freedom, the two 7-DoF arms and the two 20-DoF hands, for bimanual tabletop manipulation. The torso and neck stay at fixed joint positions, and the mobile base is not used. A single Stereolabs ZED Mini stereo camera (Stereolabs, 2018) mounted on the head provides the visual input (Figure 9).

Observation space. At every policy query the observation has three parts. The first is two egocentric RGB images, the left and right views of the ZED Mini stereo pair, captured at 30 Hz and resized to 192×320. The second is a 62-dimensional proprioceptive vector of measured joint positions. The third is the task instruction (Table 6).

Action space. Actions are absolute joint-position targets. The base policy predicts an action chunk of 24 steps, and its first 16 steps are executed. RL acts only on these 16 executed steps and on the 54 arm and hand joints. SQAM optimizes this sub-chunk by updating the action head of the base policy, whereas EXPO-FT keeps the base policy frozen and adds a residual to it through its edit policy.

Control loop. The robot runs at a nominal control rate of 30 Hz and executes the policy synchronously. At each policy query the current observation is sent to the actor and the returned 16 step chunk of joint-position targets is executed step by step. A new observation is taken and the next chunk is requested only after the chunk has been completed.

E.2 Tasks and evaluation protocol

Table 6 lists the three tasks with their language instructions, demonstration counts, and horizon parameters. All three tasks use a sparse reward, which is 1 when the episode succeeds and 0 otherwise.

Table 6: **Task specification.** For each task we list the language instruction, the number of teleoperated demonstrations, and the discount factor γ.

Task	Language instruction	Demos	γ
`flip-plastic-bag`	"flip the plastic bag"	50	0.999
`place-straw`	"place the straw in the cup"	50	0.999
`put-and-close`	"put the fruit in the pot and close the lid"	50	0.999

Each task uses a fixed initial configuration for the teleoperated demonstrations, training, and evaluation, identical across methods. Every checkpoint is evaluated over 30 held-out episodes, and success is judged by the operator from the final state of the episode. At serving time SQAM samples one action chunk per query, while EXPO-FT serves the best of six sampled candidates per query.

Differences from the EXPO-FT recipe. We run EXPO-FT with three changes to its released recipe, so that both methods follow the same protocol. First, neither method uses the corrective actions from a human operator that the EXPO-FT recipe includes. Second, its base policy is kept frozen and is not updated with its original training objective. Third, the critic of both methods reads the same mean-pooled VLM feature in place of the separate ResNet encoder of EXPO-FT, to reduce the compute cost. The remaining settings are shared with SQAM (Table 7), and EXPO-FT uses best-of-N sampling with $N = 6$ at training and serving time.

E.3 Training details

Since SQAM updates the pretrained policy itself, its actor learning rate is small. EXPO-FT trains a residual edit policy and a Q head on top of the frozen base policy, with the changes listed in Section E.2, and its online run continues from its own 15K offline checkpoint. Its learning rates are those of EXPO-FT, and its edit scale is the smallest value that work uses. Table 7 lists all hyperparameters.

E.4 Additional results

Figure 10 tracks the success rate of the training rollouts during the online phase, and the rollout success rate of SQAM tends to increase as training proceeds on all three tasks. Figures 11 to 13 show the final frames of the evaluation episodes and the main failure mode of each method. The difference is clearest on `flip-plastic-bag`. The superimposed final frames of SFT show many episodes that end in the middle of flipping the bag, and these become visibly fewer under SQAM, consistent with its higher success rate. EXPO-FT instead often drops the bag while flipping it. Figure 14 compares the duration of the successful episodes. SQAM completes `flip-plastic-bag` and `put-and-close` faster than SFT and `place-straw` in a comparable time.

Table 7: **Hyperparameters for the real-robot runs.** The shared block applies to both methods, and the blocks below give each method's own values.

Parameter	Value
Shared	
Offline RL steps	15,000
Online RL steps	10,000
Batch size	128 offline, 192 online
Policy delay	5
UTD ratio	1
Action horizon	16
Action DoF	54
Critic ensemble size (K)	10
Network width	512
Network depth	4 hidden layers
Critic input	mean-pooled VLM feature, proprioception, action chunk
SQAM	
Actor / critic learning rate	10^{-5} / 10^{-4}
Value penalty coefficient (c)	0.01
KL budget ($\varepsilon_{\mathrm{KL}}$)	0.5
EXPO-FT	
Actor / critic learning rate	3×10^{-4} / 3×10^{-4}
Best-of-N sampling (training and serving)	6
Edit scale	0.05

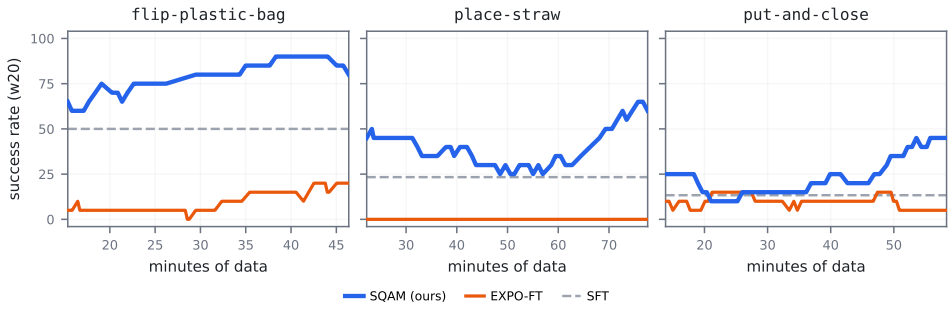
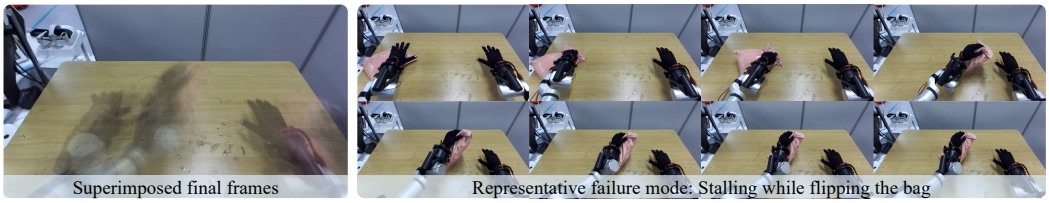
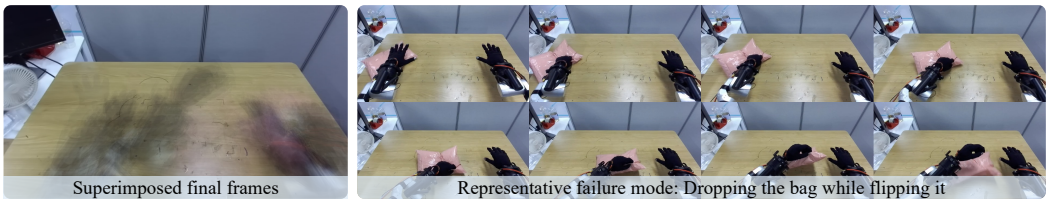


Figure 10: Success rate of the training rollouts during the online RL phase, over a trailing window of 20 episodes, against minutes of robot data collected. The dashed line is the SFT policy's held-out success rate. SQAM works at the scale of a vision-language-action policy, improving over SFT on all three tasks.

(a) SFT



(b) Expo-FT



(c) SQAM

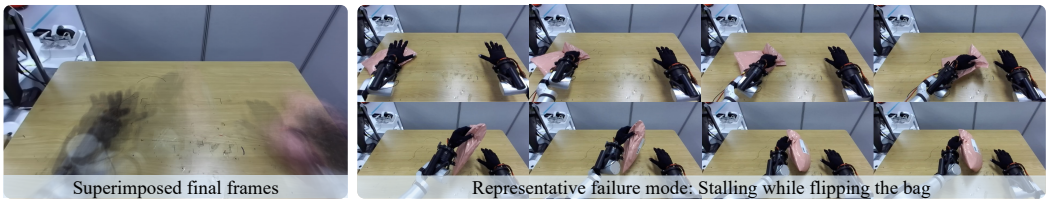
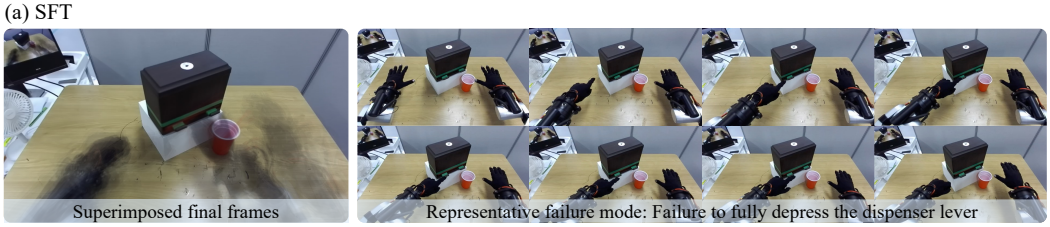


Figure 11: Evaluation episodes on `flip-plastic-bag`. Left: superimposed final frames of all episodes. Right: the main failure mode of each method.

(a) SFT



(b) Expo-FT



(c) SQAM



Figure 12: Evaluation episodes on `place-straw`. Left: superimposed final frames of all episodes. Right: the main failure mode of each method.

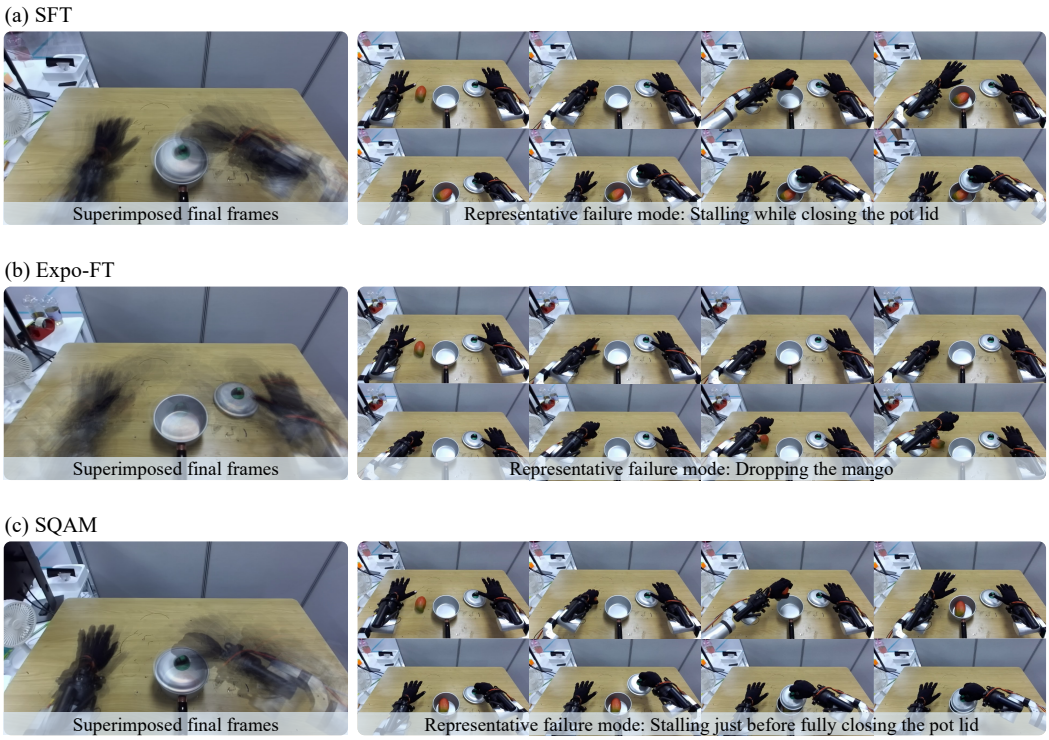


Figure 13: Evaluation episodes on `put-and-close`. Left: superimposed final frames of all episodes. Right: the main failure mode of each method.

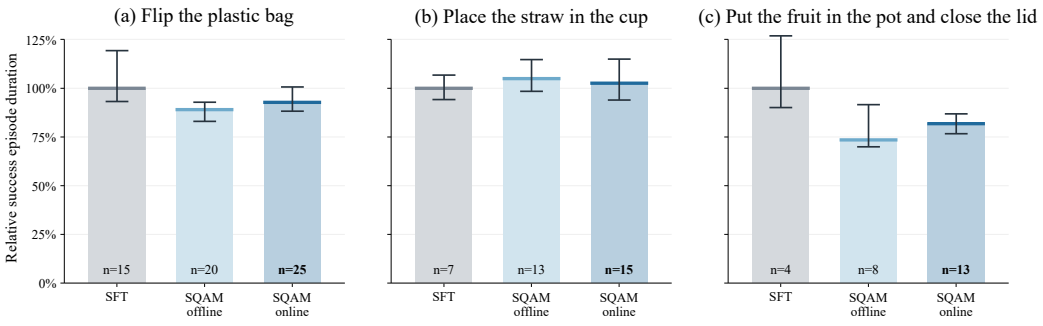


Figure 14: Success episode durations normalized by each task's SFT median (100%). Bars show medians, error bars indicate the 25th–75th percentiles, and n denotes the number of success episodes.

F Additional experiments

F.1 The cost of one update

To measure the compute that the scalar adjoint saves, Figure 15 compares the wall-clock time of one SQAM update with that of the same update using the exact adjoint. The exact adjoint takes a vector–Jacobian product through the policy at each of the F flow steps, and the scalar adjoint takes none. The left panel widens the actor with the critic fixed. The exact adjoint remains about half again as slow at every width, so the saving persists on larger policies. The right panel varies the number of flow steps F. The gap grows with F, since the cost of the exact adjoint grows with the number of vector–Jacobian products while the rest of the update does not.

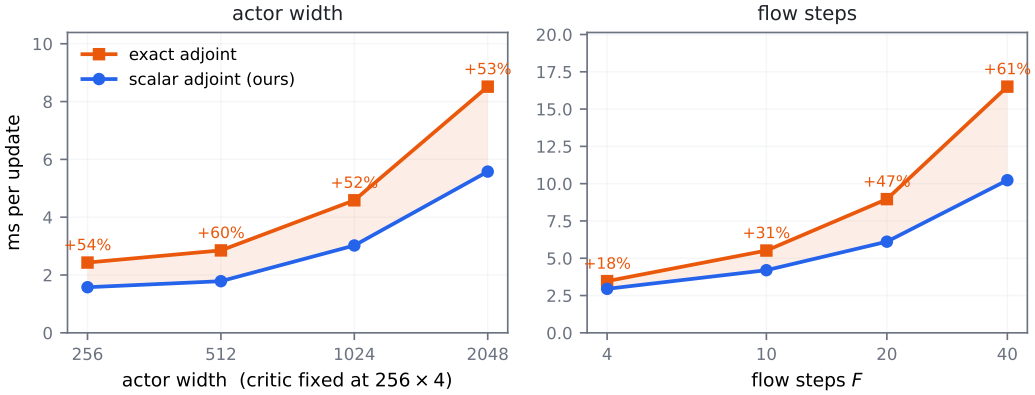


Figure 15: Wall-clock time of one update with the exact and the scalar adjoint. **Left:** actor width, with the critic fixed at 256×4. **Right:** number of flow steps F. Percentages give the extra time of the exact adjoint.

F.2 The value penalty inside TRQAM

Section 4.3 explains the effect of the value penalty by where the scalar adjoint uses the critic, which is its gradient at the policy-generated action, used directly at every flow step. The exact adjoint instead passes this gradient through the velocity Jacobian at every flow step, so if the explanation holds, the value penalty should help the exact adjoint less. To test this, we switch the penalty on and off in both TRQAM and SQAM on `cube-triple` and `cube-quadruple`, the two domains where the scalar adjoint alone underperforms. All four configurations use the same four seeds with the trust region active, and they differ only in the critic loss.

Figure 16 shows that the value penalty improves TRQAM as well, but far less than it improves SQAM. At the offline endpoint, it raises TRQAM by 18 points on `cube-triple` and 23 on `cube-quadruple`, and raises SQAM by 48.5 and 78 points. This matches the prediction above.

F.3 Without the trust region

Section 3.1 notes that SQAM can be used with or without the path-space trust region of TRQAM. To run it without, we freeze the dual variable, $\eta_\lambda = 0$, so that λ stays at a constant value as in QAM. Figure 17 runs five constants $\lambda \in \{0.1, 0.3, 1, 3, 10\}$ on one representative task from each of the ten domains, with the other settings, including the value penalty coefficients, as in the main experiments and eight seeds per curve. The dashed line is SQAM with the trust region.

Without the trust region, the effect of λ is less predictable across domains. The best constant differs by two orders of magnitude, for example $\lambda = 0.1$ is the best value on `antmaze-giant` but fails on `puzzle-3x3`, while $\lambda = 10$ is the best value on `puzzle-3x3` but fails on `antmaze-large`. With the trust region, the KL budget affects performance more predictably (Section 4.4) and keeps training stable, so we use it by default.

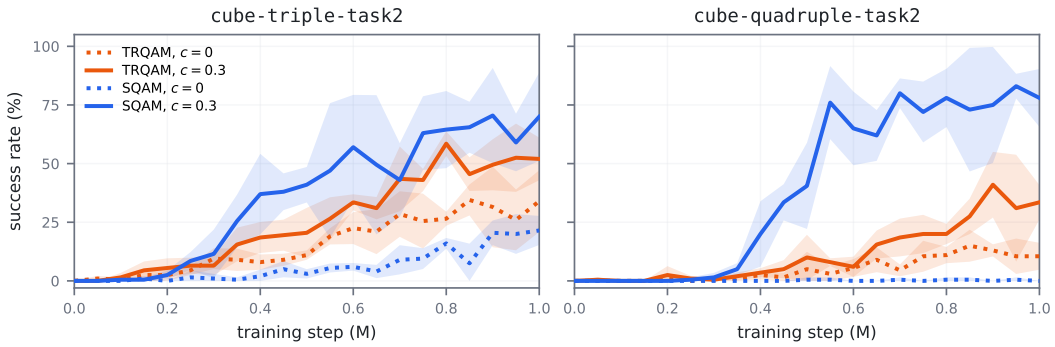


Figure 16: TRQAM and SQAM with and without the value penalty ($c = 0.3$ and $c = 0$) during offline training on task 2 of `cube-triple` and `cube-quadruple`. Four seeds per curve, with the trust region at the KL budget of each method.

F.4 Batch-averaged velocity Jacobian on all domains

Figures 18 and 19 extend Figure 2 to all ten domains at four flow times. As in Figure 2, the velocity Jacobian of the pretrained flow is averaged over a batch of states sampled from the dataset and normalized by its mean diagonal. On every domain, the batch-averaged Jacobian is dominated by its diagonal.

F.5 Ablation studies on all domains

Figures 20 and 21 extend the ablations of Section 4.4 to all ten domains, with one representative task per domain and eight seeds per setting, offline to 1M steps and online to 1.5M. They sweep the KL budget and the value penalty coefficient c. The trends of the main text hold across the ten domains. A small KL budget performs best or equally well on most domains. The exception is `puzzle-4x4`, which prefers a larger budget, consistent with its larger state space and with the budget that Dong et al. (2026c) report for it. As in Section 4.4, the value penalty matters mainly on the domains where the scalar adjoint alone underperforms, with large gains on `cube-triple` and `cube-quadruple` and little gain on most other domains. On `cube-double`, it lowers the offline success rate at $c = 0.3$, so the main experiments run this domain without the penalty.

F.6 Value learning schemes on all domains

Figures 22 to 24 extend the comparison of value learning schemes in Section 4.3 to all ten domains, with the same tasks, seeds, and steps as Section F.5. They sweep the gradient-norm penalty, ReBRAC (Tarasov et al., 2023), and the in-sample critic (Kostrikov et al., 2022) over their own coefficients, and the unregularized critic (*none*) is drawn as a dashed reference. Consistent with Section 4.3, the gradient-norm penalty also helps, most visibly on `puzzle-3x3` and `cube-quadruple`, but degrades several domains at its largest coefficient, ReBRAC stays close to the unregularized critic except at its largest β, and the in-sample critic falls below the unregularized critic on most domains. The value penalty sweep of Figure 21 completes the comparison.

G Full OGBench results

Table 8 reports the offline success rate of every method on all fifty tasks at 1M steps, the per-task breakdown behind Table 1. Figures 25 and 26 show the corresponding success rate curves, 1M offline steps followed by 0.5M online, with eight seeds per method. SQAM uses the hyperparameters of Section D.2 on every task.

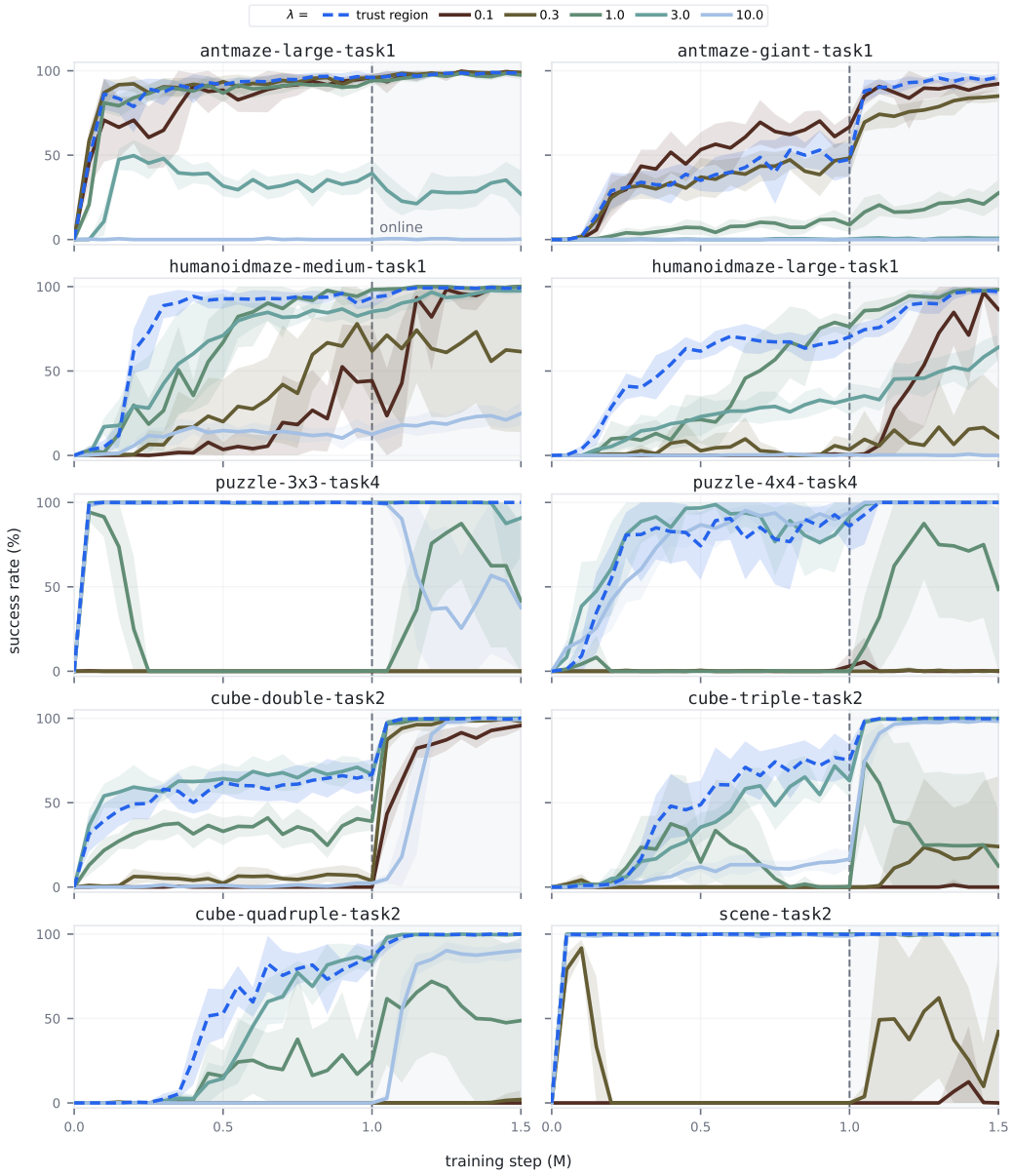


Figure 17: SQAM with a constant λ, for five values, against SQAM with the trust region (dashed) on one representative task from each of the ten domains. Eight seeds per curve, and bands are one standard deviation.

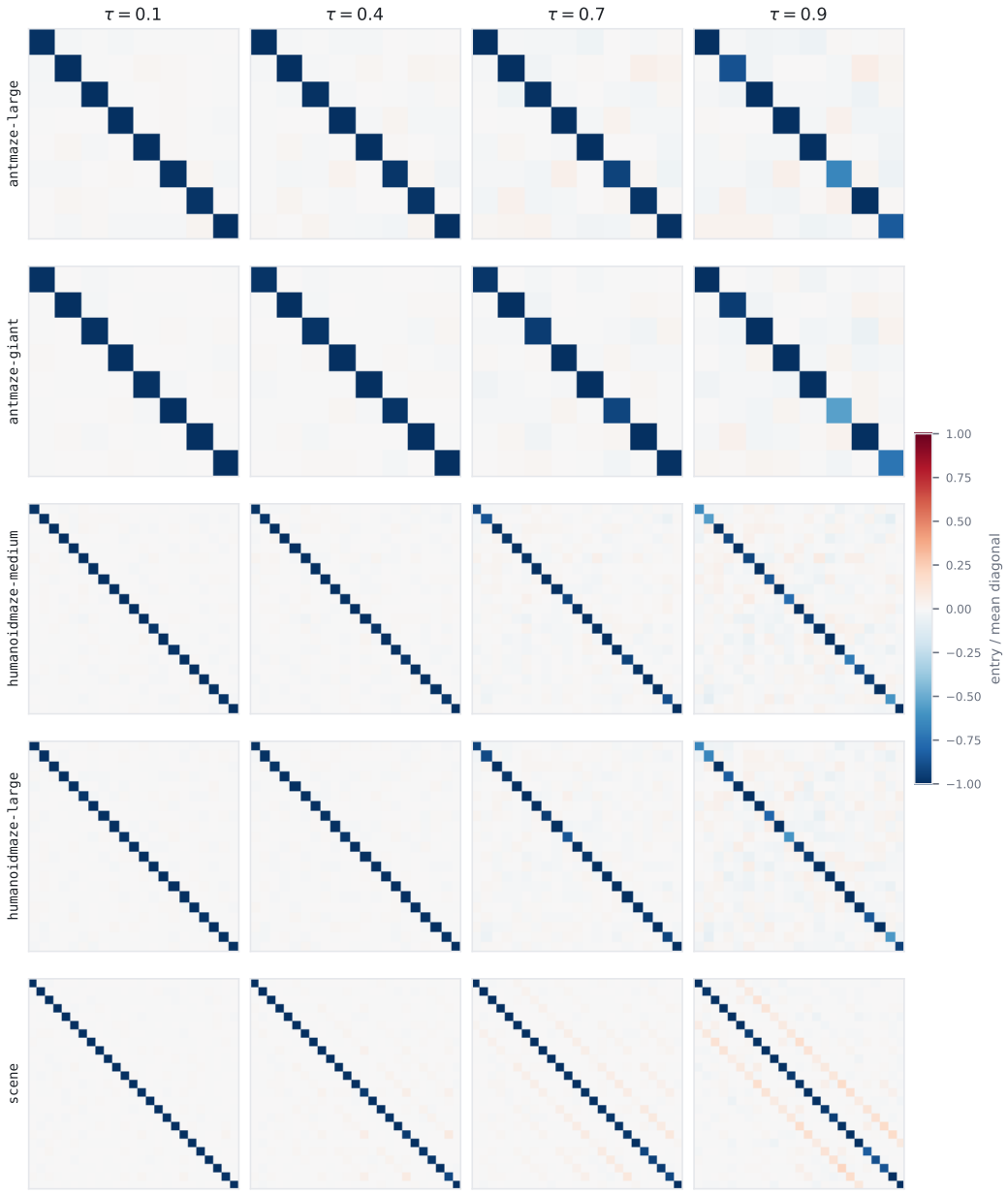


Figure 18: The batch-averaged velocity Jacobian on the locomotion domains and `scene` at four flow times, each panel normalized by its mean diagonal.

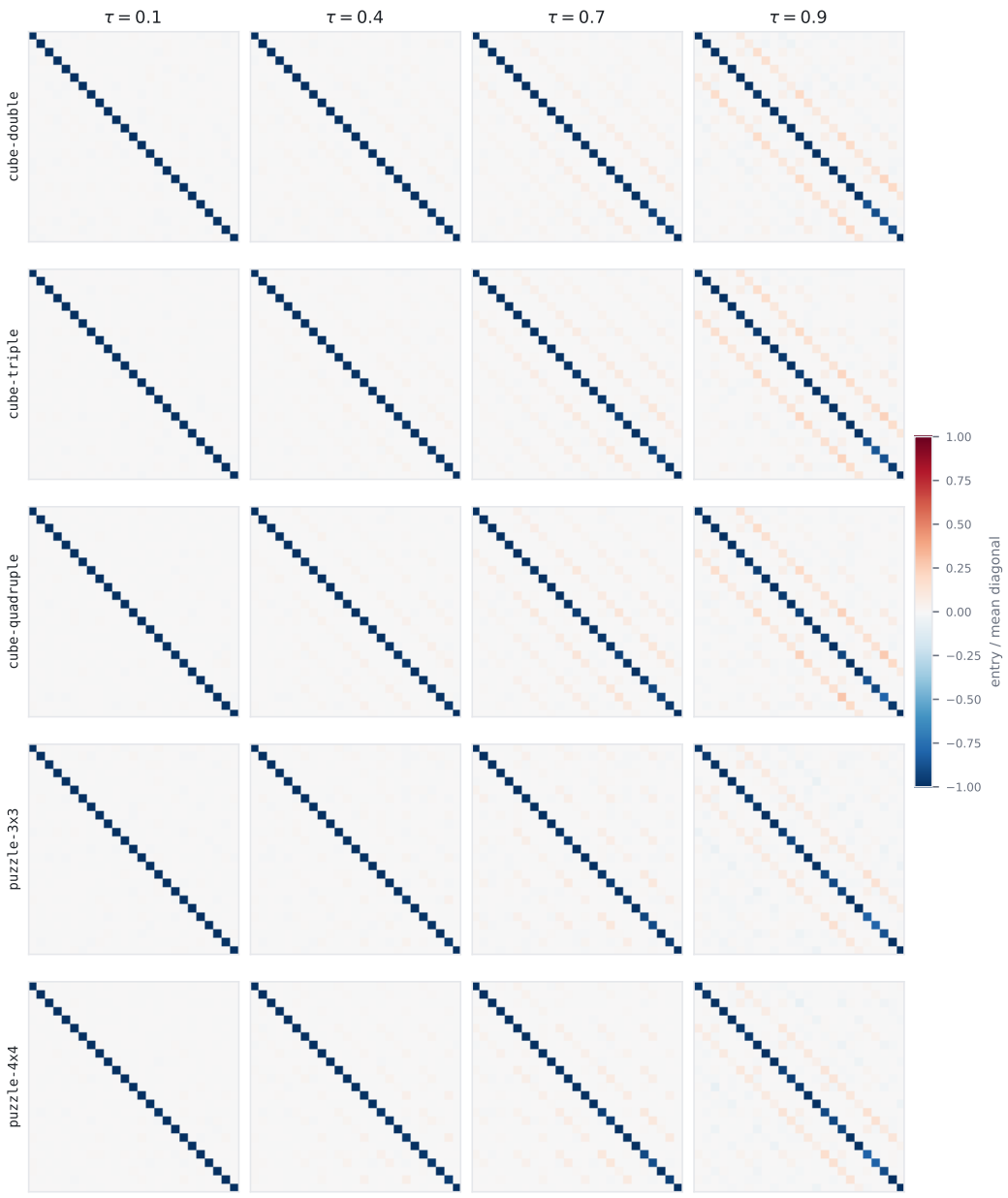


Figure 19: The batch-averaged velocity Jacobian on the `cube` and `puzzle` domains at four flow times, with the same normalization as Figure 18.

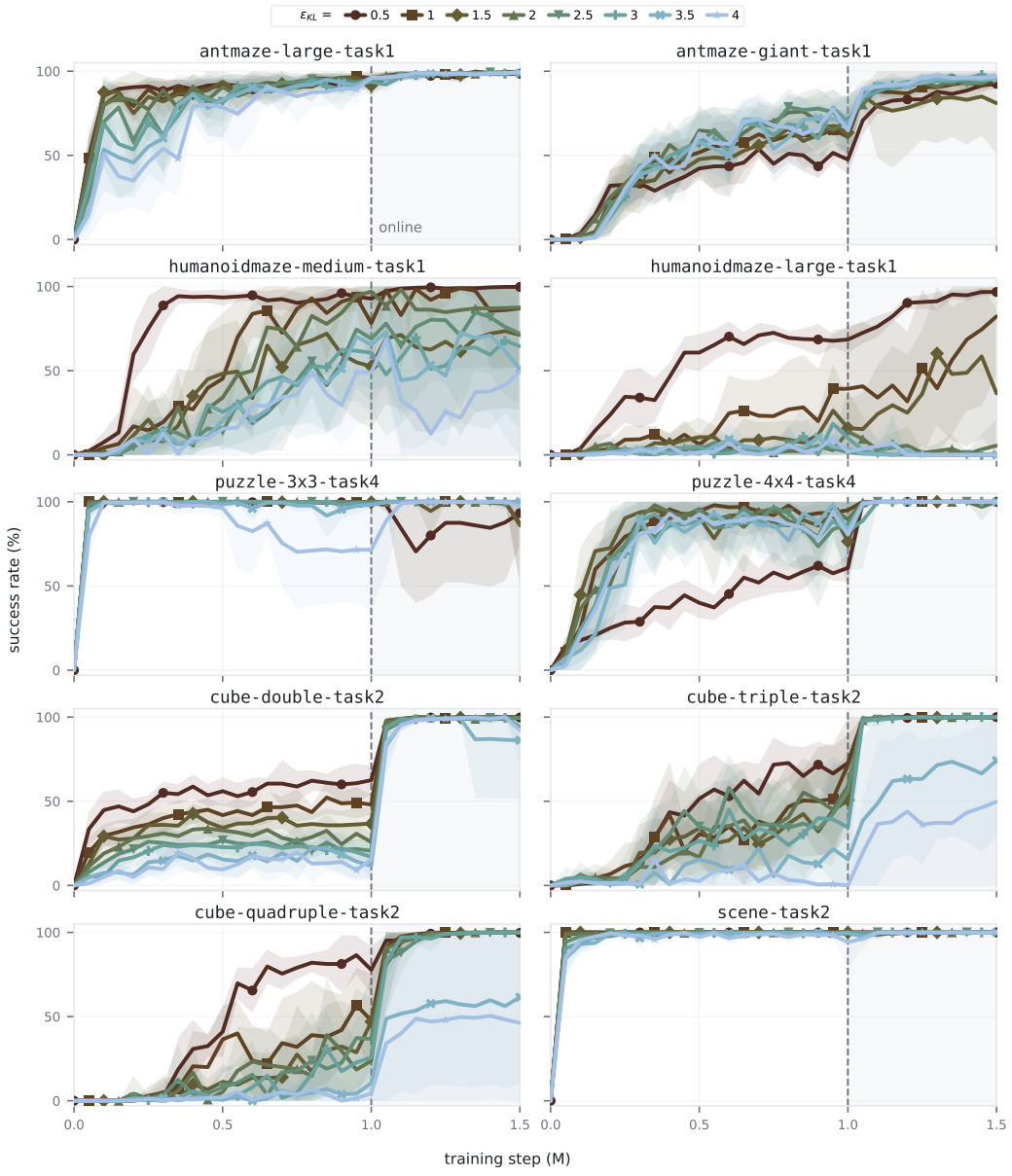


Figure 20: The KL budget $\varepsilon_{\mathrm{KL}}$ swept from 0.5 to 4 on all ten domains. Eight seeds per budget, and bands are one standard deviation.

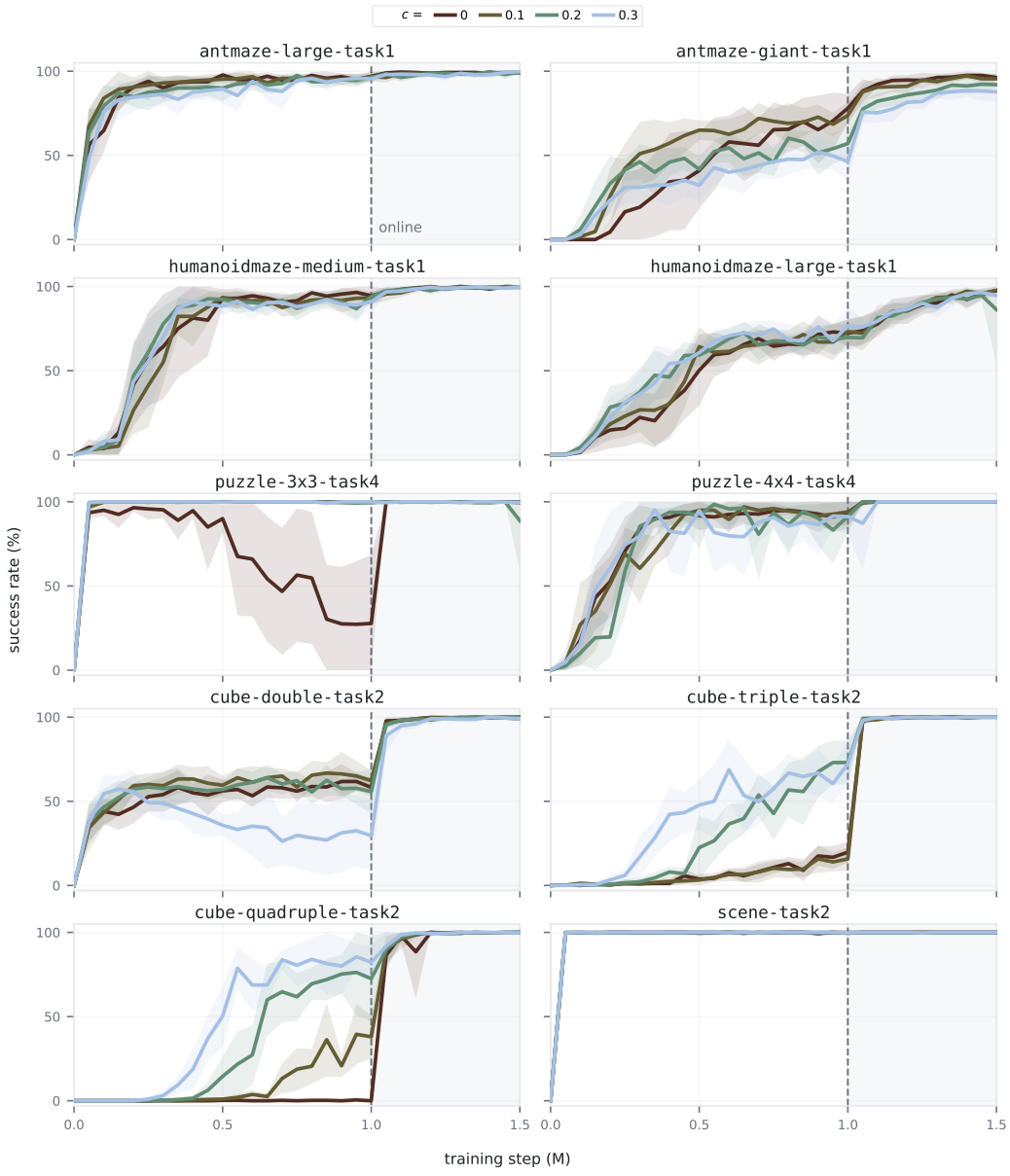


Figure 21: The value penalty coefficient c swept over $\{0, 0.1, 0.2, 0.3\}$ on all ten domains. Eight seeds per coefficient, and bands are one standard deviation.

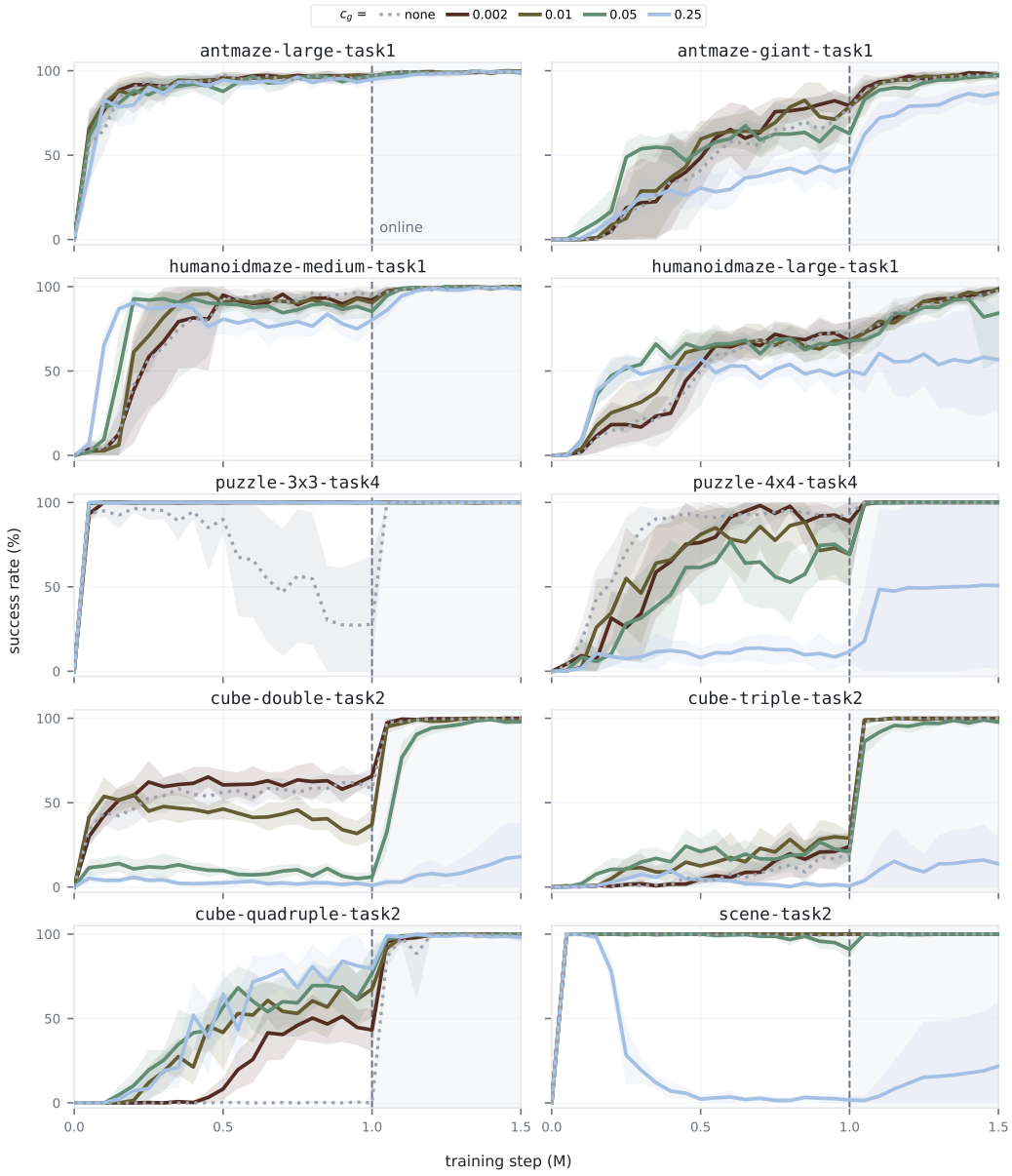


Figure 22: The gradient-norm penalty coefficient c_g swept over $\{0.002, 0.01, 0.05, 0.25\}$ on all ten domains, with the unregularized critic (*none*) as the dashed reference. Eight seeds per coefficient, and bands are one standard deviation.

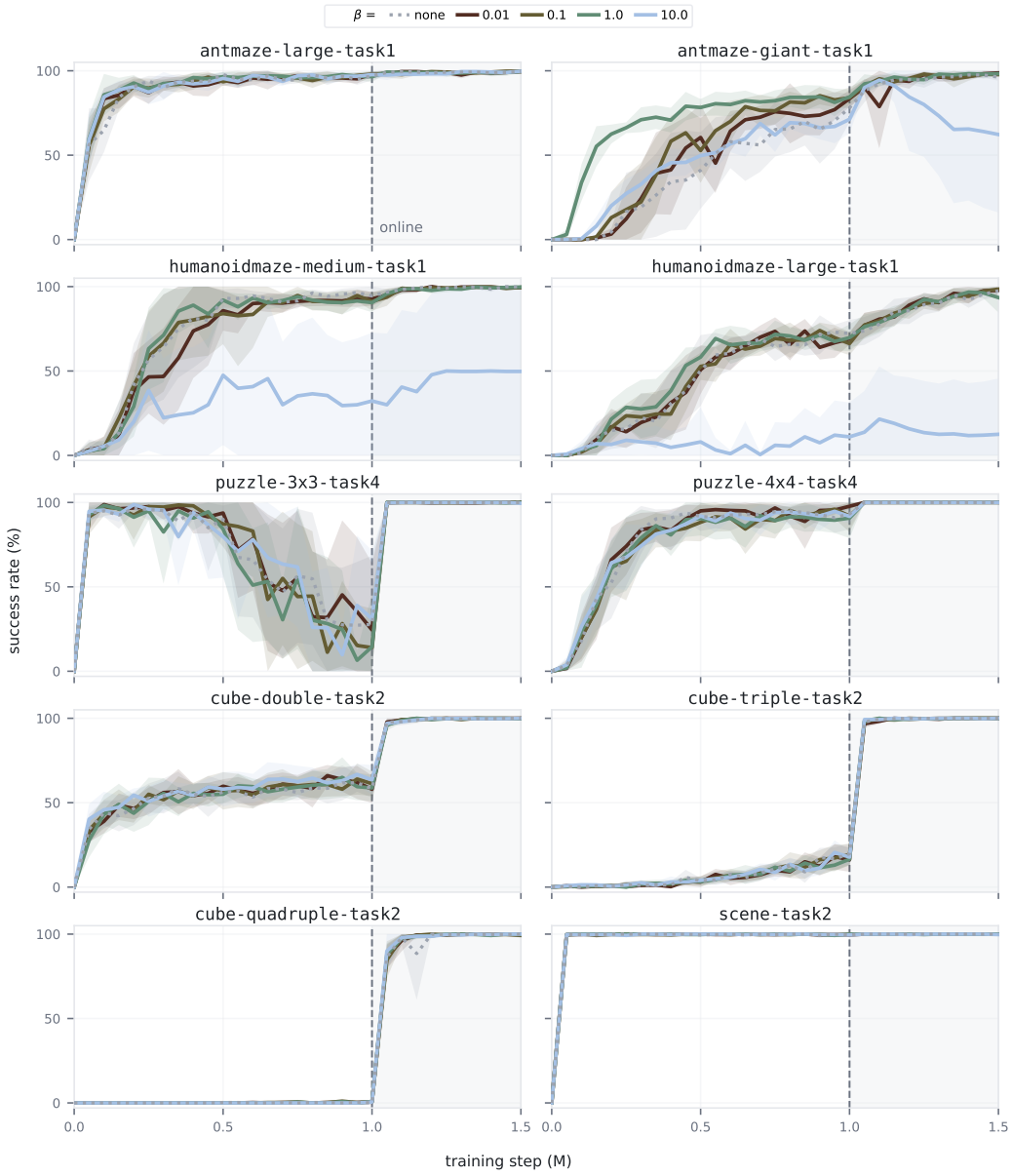


Figure 23: ReBRAC's bootstrap penalty swept over $\beta \in \{0.01, 0.1, 1, 10\}$ on all ten domains, with the unregularized critic (*none*) as the dashed reference. Eight seeds per coefficient, and bands are one standard deviation.

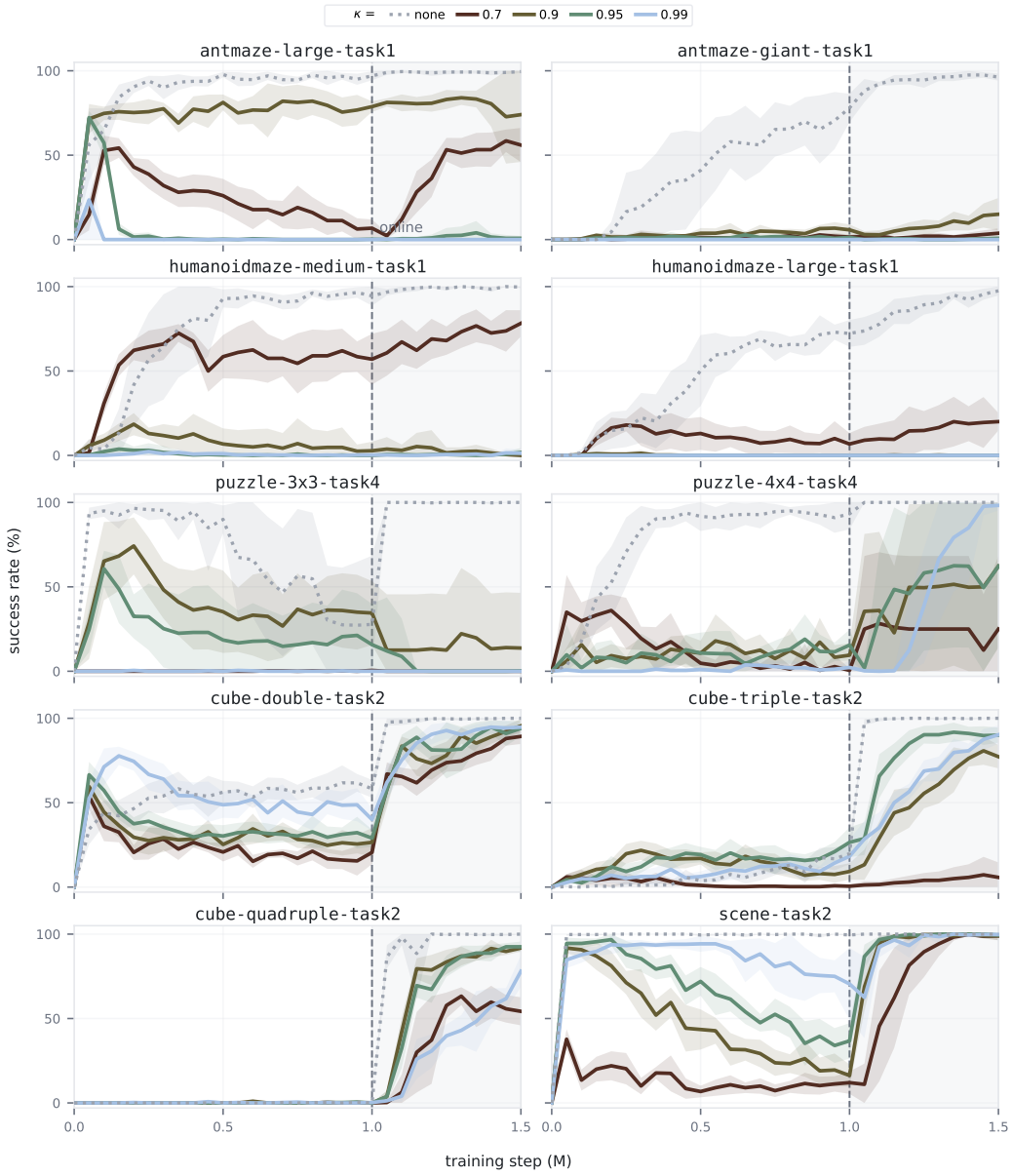


Figure 24: The in-sample critic swept over the expectile $\kappa \in \{0.7, 0.9, 0.95, 0.99\}$ on all ten domains, with the unregularized critic (*none*) as the dashed reference. Eight seeds per expectile, and bands are one standard deviation.

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

		FQL	CGQL-L	DSRL	IFQL	QAM	QAM-E	TRQAM	SQAM
antmaze-large	task1	24 ±44	62 ±38	45 ±12	7 ±11	96 ±3	93 ±5	95 ±4	**96** ±2
	task2	0 ±0	21 ±36	73 ±7	9 ±14	36 ±49	91 ±4	87 ±6	85 ±4
	task3	73 ±14	88 ±5	84 ±5	62 ±24	86 ±7	93 ±3	93 ±3	**93** ±4
	task4	0 ±0	0 ±0	0 ±0	35 ±22	0 ±0	66 ±9	75 ±17	**88** ±7
	task5	92 ±4	68 ±42	63 ±8	34 ±24	94 ±3	89 ±5	96 ±3	93 ±3
	agg. (5 tasks)	38 ±9	48 ±7	53 ±2	29 ±8	62 ±9	86 ±3	89 ±4	**91** ±3
antmaze-giant-10M	task1	1 ±1	11 ±12	2 ±3	0 ±0	37 ±9	0 ±0	48 ±5	**48** ±11
	task2	0 ±0	14 ±12	0 ±0	43 ±13	18 ±19	0 ±0	21 ±12	**86** ±3
	task3	0 ±0	0 ±0	0 ±0	14 ±8	2 ±2	0 ±0	5 ±5	**30** ±10
	task4	0 ±0	0 ±0	0 ±0	1 ±2	20 ±24	0 ±0	58 ±13	**61** ±8
	task5	10 ±29	8 ±23	2 ±2	2 ±2	69 ±8	28 ±38	75 ±7	**83** ±8
	agg. (5 tasks)	2 ±6	7 ±5	1 ±1	12 ±3	29 ±4	6 ±8	41 ±4	**62** ±4
humanoidmaze-medium	task1	84 ±21	84 ±9	24 ±23	93 ±3	33 ±34	28 ±30	87 ±5	**94** ±6
	task2	99 ±1	99 ±1	88 ±6	94 ±4	100 ±1	98 ±2	89 ±7	96 ±3
	task3	89 ±9	0 ±0	64 ±27	96 ±4	90 ±8	74 ±18	90 ±6	**98** ±2
	task4	0 ±0	0 ±0	0 ±0	82 ±8	0 ±0	0 ±0	61 ±7	**82** ±7
	task5	99 ±1	100 ±1	88 ±5	99 ±2	100 ±1	99 ±1	95 ±4	**100** ±1
	agg. (5 tasks)	74 ±5	57 ±2	53 ±10	93 ±2	64 ±7	60 ±6	84 ±3	**94** ±2
humanoidmaze-large	task1	1 ±2	16 ±11	1 ±1	32 ±5	6 ±10	9 ±14	45 ±8	**70** ±7
	task2	0 ±0	0 ±0	0 ±0	6 ±9	0 ±0	0 ±0	10 ±6	**25** ±13
	task3	7 ±4	16 ±8	2 ±3	76 ±6	7 ±5	3 ±4	34 ±15	57 ±17
	task4	0 ±0	0 ±1	0 ±0	35 ±26	0 ±0	0 ±0	44 ±13	**60** ±14
	task5	0 ±1	0 ±0	1 ±1	0 ±0	5 ±11	9 ±18	45 ±6	**59** ±14
	agg. (5 tasks)	2 ±1	6 ±3	1 ±1	30 ±7	4 ±3	4 ±5	36 ±4	**54** ±5
scene	task1	100 ±0	100 ±0	100 ±0	99 ±1	100 ±0	100 ±0	100 ±0	**100** ±0
	task2	100 ±1	99 ±1	100 ±0	2 ±2	100 ±0	100 ±0	100 ±0	**100** ±1
	task3	82 ±5	93 ±7	100 ±1	77 ±7	94 ±4	93 ±4	100 ±1	**99** ±2
	task4	60 ±22	0 ±0	100 ±0	2 ±2	26 ±21	22 ±30	93 ±6	**99** ±2
	task5	8 ±7	0 ±0	0 ±0	0 ±0	0 ±0	0 ±0	0 ±1	0 ±0
	agg. (5 tasks)	70 ±5	58 ±1	80 ±0	36 ±1	64 ±4	63 ±6	79 ±1	**79** ±0
puzzle-3x3	task1	87 ±11	0 ±0	100 ±0	100 ±1	75 ±15	99 ±2	100 ±0	**100** ±0
	task2	37 ±39	0 ±0	100 ±0	23 ±7	0 ±0	100 ±0	100 ±0	**100** ±0
	task3	0 ±0	0 ±0	100 ±0	82 ±11	0 ±0	76 ±11	100 ±0	**100** ±1
	task4	0 ±0	0 ±0	100 ±0	29 ±8	0 ±0	87 ±6	100 ±1	**100** ±1
	task5	3 ±6	0 ±0	100 ±0	86 ±9	0 ±1	84 ±20	100 ±0	**100** ±0
	agg. (5 tasks)	25 ±10	0 ±0	100 ±0	64 ±4	15 ±3	89 ±4	100 ±0	**100** ±0
puzzle-4x4-10M	task1	3 ±3	1 ±1	90 ±5	74 ±11	3 ±4	90 ±10	100 ±0	**97** ±6
	task2	6 ±9	0 ±0	9 ±9	4 ±3	1 ±1	43 ±22	99 ±1	73 ±30
	task3	23 ±15	0 ±0	81 ±20	84 ±6	2 ±3	57 ±10	98 ±2	81 ±12
	task4	7 ±5	0 ±0	76 ±28	46 ±13	0 ±1	31 ±16	100 ±1	86 ±16
	task5	9 ±16	0 ±0	49 ±43	3 ±1	1 ±1	50 ±22	98 ±5	90 ±17
	agg. (5 tasks)	9 ±7	0 ±0	61 ±8	42 ±4	1 ±1	54 ±8	99 ±1	85 ±6
cube-double	task1	75 ±10	83 ±4	85 ±7	18 ±7	96 ±3	94 ±4	98 ±2	94 ±5
	task2	59 ±11	66 ±6	85 ±5	9 ±2	81 ±6	82 ±7	92 ±3	67 ±10
	task3	39 ±8	72 ±7	82 ±10	8 ±4	75 ±7	67 ±8	80 ±12	61 ±10
	task4	6 ±3	15 ±5	35 ±6	2 ±2	24 ±6	29 ±6	54 ±8	9 ±3
	task5	44 ±9	37 ±6	71 ±4	7 ±3	78 ±6	82 ±6	82 ±6	61 ±11
	agg. (5 tasks)	44 ±4	55 ±2	72 ±4	9 ±2	71 ±2	71 ±3	81 ±3	58 ±3
cube-triple-10M	task1	26 ±20	2 ±3	70 ±18	51 ±22	56 ±22	26 ±11	87 ±8	**98** ±4
	task2	1 ±1	0 ±0	16 ±7	16 ±3	15 ±11	14 ±9	42 ±9	**75** ±10
	task3	8 ±6	0 ±0	40 ±8	28 ±8	21 ±11	9 ±6	45 ±9	**81** ±11
	task4	2 ±3	0 ±0	15 ±5	1 ±1	4 ±3	3 ±3	30 ±13	**68** ±10
	task5	1 ±1	0 ±0	31 ±12	23 ±8	1 ±1	2 ±4	48 ±17	37 ±10
	agg. (5 tasks)	7 ±5	0 ±1	34 ±6	24 ±7	19 ±6	11 ±4	50 ±5	**72** ±4
cube-quadruple-100M	task1	36 ±25	7 ±5	36 ±12	15 ±11	70 ±10	45 ±19	66 ±16	**90** ±6
	task2	4 ±5	0 ±1	5 ±3	5 ±7	2 ±4	0 ±0	18 ±11	**87** ±6
	task3	3 ±3	0 ±0	2 ±2	6 ±8	16 ±11	2 ±4	11 ±7	**62** ±8
	task4	0 ±0	0 ±0	0 ±1	0 ±0	2 ±2	0 ±0	3 ±2	**31** ±17
	task5	0 ±0	0 ±0	1 ±1	2 ±3	0 ±0	0 ±0	0 ±0	0 ±0
	agg. (5 tasks)	9 ±5	1 ±1	9 ±3	6 ±3	18 ±3	9 ±3	19 ±5	**54** ±5
all	agg. (50 tasks)	28	23	46	35	35	45	68	**75**

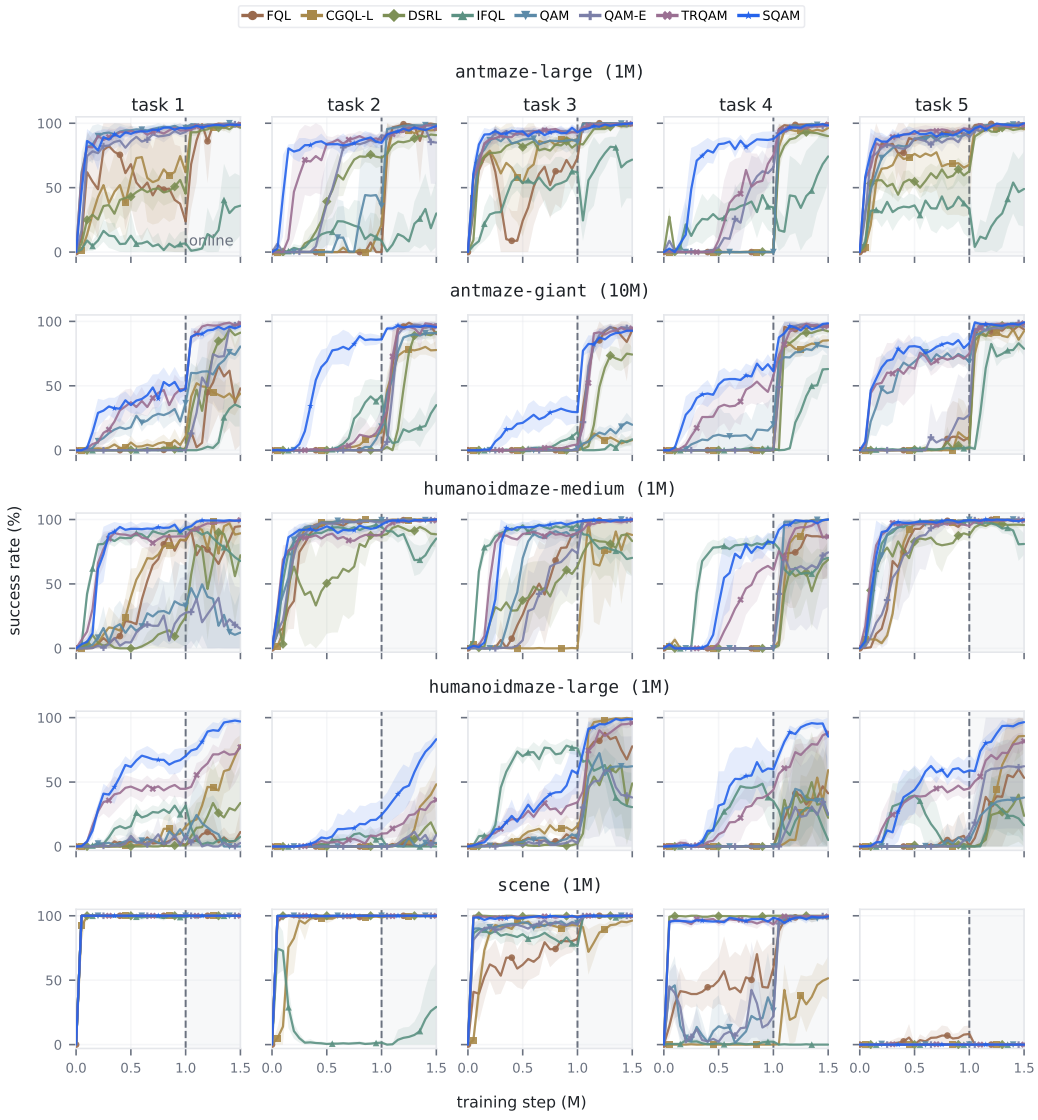


Figure 25: Success rate curves on the locomotion domains and `scene`, with rows for domains and columns for their five tasks. Offline training runs to 1M steps and online training to 1.5M, with the dashed line at the transition. Eight seeds per curve, and bands are one standard deviation.

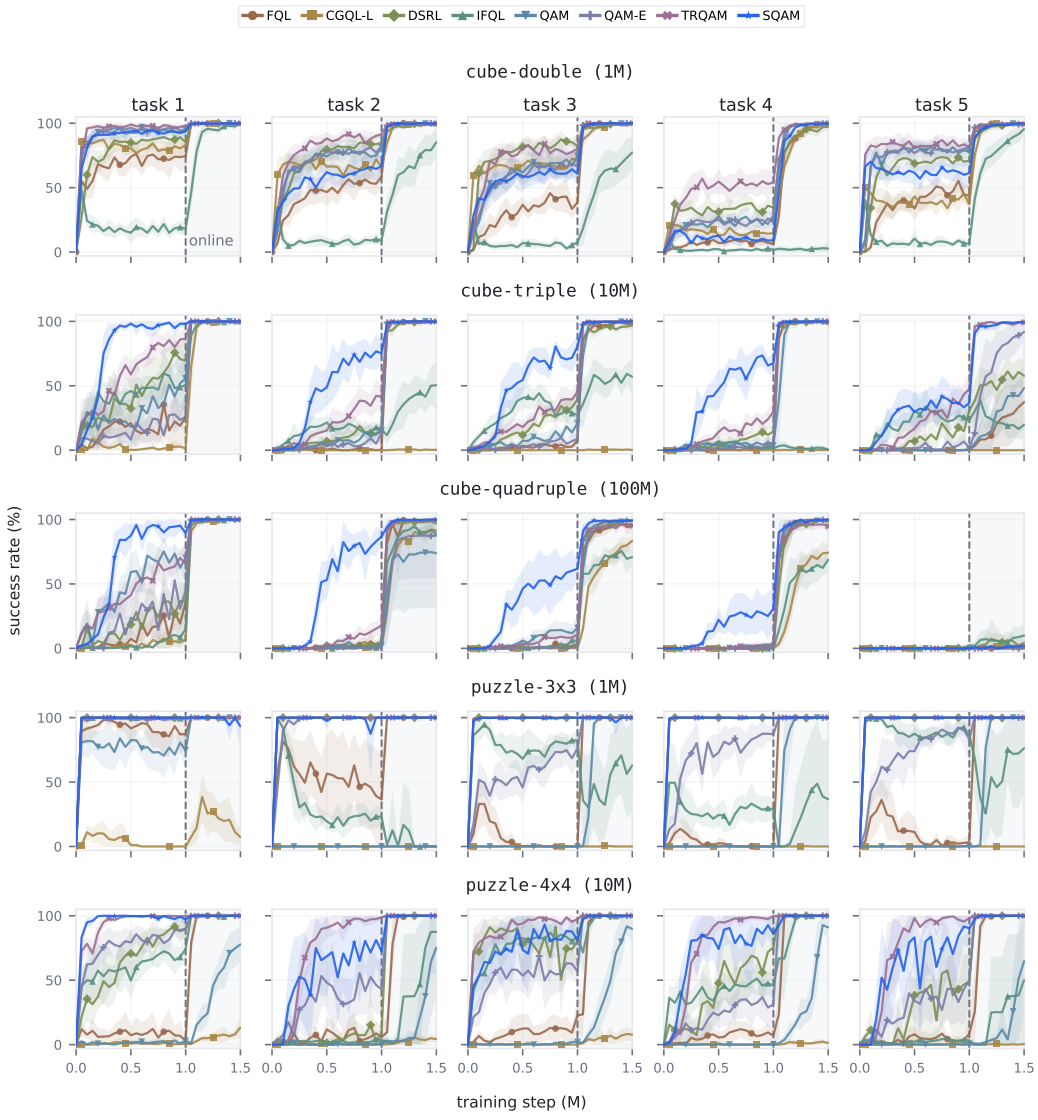


Figure 26: Success rate curves on the `cube` and `puzzle` domains, with rows for domains and columns for their five tasks. Offline training runs to 1M steps and online training to 1.5M, with the dashed line at the transition. Eight seeds per curve, and bands are one standard deviation.

H LIMITATIONS AND FUTURE WORK

We see three directions that this work leaves open.

Value learning for flow policies. Our results suggest that the value learning scheme largely determines how well the scalar adjoint performs, and SQAM addresses this with a value penalty added to a standard critic. A value learning method designed for flow policies from the ground up may make this dependence more reliable than a penalty on top of an existing critic.

Multi-task fine-tuning. Updating the generative policy itself, rather than a separately learned component, should matter most when a single policy serves many tasks. Our real-robot experiments fine-tune the policy on each task separately. Fine-tuning a single policy across many tasks, as in Wang et al. (2026), would test this benefit more directly.

Why the Jacobian concentrates on its diagonal. We observe that the batch-averaged velocity Jacobian of pretrained flow policies concentrates on its diagonal on every domain we measured, but we do not explain why. A theoretical analysis of when this structure arises would clarify the conditions under which the scalar adjoint applies.