Aligning Flow Map Policies with Optimal Q-Guidance

Christos Ziakas[1]* **Alessandra Russo**[1] **Avishek Joey Bose**[1, 2]

[1]Imperial College London [2]Mila

Abstract

Generative policies based on expressive model classes, such as diffusion and flow matching, are well-suited to complex control problems with highly multimodal action distributions. Their expressivity, however, comes at a significant inference cost: generating each action typically requires simulating many steps of the generative process, compounding latency across sequential decision-making rollouts. We introduce *flow map policies*, a novel class of generative policies designed for fast action generation by learning to take arbitrary-size jumps—including one-step jumps—across the generative dynamics of existing flow-based policies. We instantiate flow map policies for offline-to-online reinforcement learning (RL) and formulate online adaptation as a trust-region optimization problem that improves the critic's Q-value while remaining close to the offline policy. We theoretically derive FLOW MAP Q-GUIDANCE (FMQ), a principled closed-form learning target that is optimal for adapting offline flow map policies under a critic-guided trust-region constraint. We further introduce Q-GUIDED BEAM SEARCH (QGBS), a stochastic flow-map sampler that combines renoising with beam search to enable iterative inference-time refinement. Across 12 challenging robotic manipulation and locomotion tasks from OGBench and RoboMimic, FMQ achieves state-of-the-art performance in offline-to-online RL, outperforming the previous one-step policy MVP by a relative improvement of 21.3% on the average success rate.

1 Introduction

The supreme promise of offline reinforcement learning (RL) is that effective policies can be bootstrapped in a scalable data-driven manner without costly environment interaction [Levine et al., 2020]. This scaling philosophy is central to modern *data-driven* reinforcement learning [Kumar, 2019, Fu et al., 2020] that utilizes ever-growing diverse offline datasets [Collaboration et al., 2023], and now powers learning policies in high-impact applications from dialogue [Jaques et al., 2019] to robotic navigation [Kahn et al., 2018]. Indeed, imitating the highly multi-modal action distribution of expert behavior policies in such complex control problems necessitates the use of expressive policy classes that go beyond restrictive unimodal Gaussian actors [Zhu et al., 2023, Wang et al., 2022].

Generative policies based on dynamic mass transport, such as diffusion models [Sohl-Dickstein et al., 2015, Song et al., 2020] and flow-matching [Liu et al., 2022, Lipman et al., 2022, Albergo et al., 2023], provide a compelling alternative to Gaussian policies as they learn to map a simple base distribution into a rich state-conditioned action distribution [Chi et al., 2025]. The expressivity gains of generative policies make them particularly favorable for offline and offline-to-online RL [Fujimoto and Gu, 2021, Tarasov et al., 2023], where the policy must first model diverse behaviors from a static dataset and then improve through interaction. However, the price of expressive generative policies is computationally expensive inference-time simulation. More precisely, generating actions requires

*Correspondence to c.ziakas24@imperial.ac.uk

Preprint.

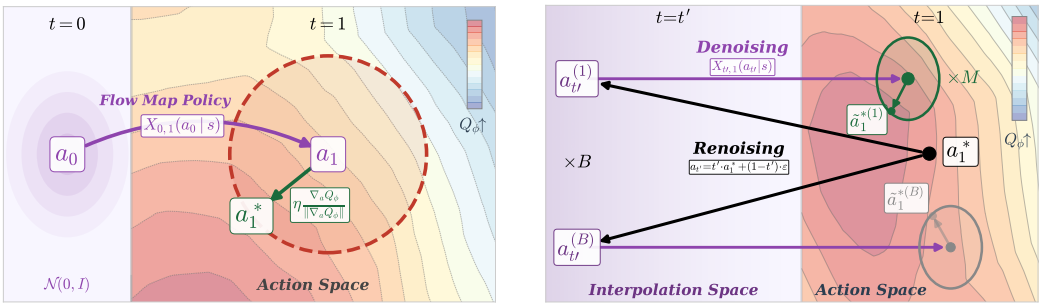

Figure 1: (Left) **FMQ**: one-step flow map policy transports noise a_0 to action a_1; then, trust-region projection displaces action a_1 to a_1^* that maximizes Q-value. (Right): **QGBS** (M=1, B=2): renoising corrupts a_1^* into B intermediate states $a_{t'}$, which the flow map policy then denoises to generate B candidate actions; candidates are updated via the optimal trust-region displacement to maximize Q_ϕ, and the highest-valued M actions are selected.

numerically integrating dynamics from noise to action and is executed at every environment step [Yang et al., 2023]—inhibiting deployment in online and real-world settings [Zhan et al., 2024, 2025].

Addressing the inference latency of generative policies is of critical interest in order to extract maximal utility from generative policies, with several recent efforts attempting to learn one-step offline policies [Park et al., 2025, Zhan et al., 2026]. However, in the context of offline-to-online RL, one-step generation remains insufficient as the policy must improve beyond the offline behavioral prior. In addition, current one-step generative policies commonly rely on heuristic-based generate-and-select procedures, such as best-of-N sampling, to bias actions toward high critic Q-values. Critically, this heuristic approach offsets the computational advantage of one-step policies by requiring many policy and critic evaluations per decision, while simultaneously not guaranteeing—for any finite N—*optimal local improvement* of the action sampled under the critic.

Present work. In this work, we introduce *flow map policies*, a novel class of generative policies that learn the unique two-time jump operator associated with the probability flow ODE of diffusion and flow-matching policies. Crucially, flow map policies generalize existing one-step policies for offline-to-online RL, e.g., mean velocity policies [Zhan et al., 2026], while introducing new learning objectives yielding Lagrange, Euler, and Progressive variations of flow map policies. In stark contrast to prior work, for principled online adaptation of one-step flow map policies, we formulate a trust-region optimization problem and derive an analytically optimal, closed-form method that aligns the action distribution with Q-value guidance. This yields our contribution FLOW MAP Q-GUIDANCE (FMQ), which constructs a novel self-bootstrapped learning target—depicted in fig. 1—as the projected action-gradient of the critic, and eliminates the need for distillation networks or best-of-N heuristics.

We additionally introduce a complementary inference-time search procedure at evaluation, Q-GUIDED BEAM SEARCH (QGBS) that combines stochastic sampling through renoising candidate samples to an intermediate state with Q-guided beam search around the trust region. Importantly, QGBS produces diverse refinements around high Q-value actions without costly ODE simulations or additional learning during online adaptation. We summarize our core contributions as follows:

1. **Flow map policies**. We introduce flow map policies as a framework for learning one-step policies as two-time jump operators for flow-based generative actors.
2. **Algorithms**. We introduce FMQ, which efficiently adapts flow map actors using optimal Q-guidance (Theorem 3.2) . We further introduce QGBS, a stochastic inference-time refinement algorithm that combines flow map renoising, beam selection, and trust-region Q-guidance.
3. **State-of-the-Art Performance:** Across 12 manipulation and locomotion tasks from OGBench and RoboMimic, FMQ outperforms prior SOTA offline-to-online baselines by a relative average of 21.3% while being on average $\approx 2.77\times$ more efficient during online adaptation.

2 Background and Preliminaries

Offline-to-Online RL. We consider a Markov Decision Process (MDP) [Sutton and Barto, 1998] defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, P, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^n$, $\mathcal{A} \subseteq \mathbb{R}^d$ denote continuous state-action spaces, $r(s, a)$ the reward function, $P(s'|s, a)$ the transition probability distribution, and $\gamma \in [0, 1)$

the discount factor. The objective of reinforcement learning is to train a policy $\pi(a|s)$ that maximizes the expected cumulative discounted return, $J(\pi) = \mathbb{E}_\pi\big[\sum_{\tau=0}^{\infty} \gamma^\tau r(s^\tau, a^\tau)\big]$, where τ denotes the timestep. Offline-to-online RL is a two-stage learning framework consisting of offline pre-training followed by online fine-tuning. In the offline pre-training phase, a behavioral prior policy is trained on a static dataset $\mathcal{D} = \{(s, a, r, s')\}$, providing an initialization. Subsequently, during online fine-tuning, the policy directly interacts with the environment. A popular approach is actor-critic methods, which employ an actor $\pi(a \mid s)$ and a critic $Q_\theta(s, a)$ that approximates the expected discounted return under policy π: $Q^\pi(s, a) = \mathbb{E}_{\pi,P}\left[\sum_{i=0}^{\infty} \gamma^i r(s^i, a^i) \mid s^0 = s, a^0 = a\right]$.

Flow matching polices. A generative policy $a_1 \sim \pi_1(\cdot|s)$ conditioned on state s, can be formulated as transport plan which pushes forward an easy to sample reference measure $p_0(a_0) \in \mathcal{P}(\mathbb{R}^d)$ to a desired measure of (optimal) target action $p_1(a_1) := p^*_{\text{target}}(a^*) \in \mathcal{P}(\mathbb{R}^d)$. The subscripts are indicative of a notion of time where the process evolves a (pseudo)-action from the prior at time $t = 0$, i.e. $a_0 \sim p_0$, to an action that follows the target distribution $a_1 \sim p_1(a_1)$ at time $t = 1$. We highlight that the time t associated with the transport dynamics is distinguished from τ, which is associated with the MDP. Formally, a *flow-policy* is a one-parameter diffeomorphism, conditioned on a state s, $\psi_t(\cdot|s) : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ that is the solution to the following ordinary differential equation:

$$\frac{d}{dt}\psi_t(a_t \mid s) = v_t\left(\psi_t(a_t \mid s)\right), \quad \psi_0(a_0) = a_0, \tag{1}$$

with initial conditions $\psi_0(a_0) = a_0$. Furthermore, $v_t : [0, 1] \times \mathbb{R}^d \to \mathbb{R}^d$ is a time-dependent (instantaneous) velocity field. In effect, the thesis of the generative policy problem is to learn a policy that pushes forward the base measure as follows $\pi_1(\cdot|s) := p_1 = [\psi_1(\cdot|s)]_\#(p_0)$. We highlight that ψ_1 produces a *deterministic* action while $\pi_1(\cdot|s)$ is induced *distribution* over actions at $t = 1$ by the flow-policy. To build the flow-policy, we can associate it with a conditional probability path $p_t(\cdot|z) : [0, 1] \times \mathcal{P}(\mathbb{R}^d) \to \mathcal{P}(\mathbb{R}^d)$ which is a time-indexed interpolation in probability space between two distributions p_0 and p_1. In its simplest form, the conditioning variable can be taken to the endpoints $z := (a_0, a_1)$, and a particle level interpolation that is simply a convex combination of the endpoints can be employed, i.e., $a_t = (1 - t)a_0 + ta_1$. We say ψ_t generates p_t if it pushes forward p_0 to p_1 by following the ODE in eq. (1). To learn the flow policy, it is easier to regress against a known target conditional velocity $v^*_t(a_t|z, s)$ field that generates p_t. With access to such a v^*_t, learning can proceed using the conditional flow-matching loss [Tong et al., 2023, Albergo et al., 2023, Liu et al., 2022, Lipman et al., 2022, Peluchetti, 2023], which is a simple simulation-free regression objective:

$$\mathcal{L}_{\text{CFM}} = \mathbb{E}_{t,q(z),p_t(a_t|z,s)}\|v_t(a_t \mid s) - v^*_t(a_t \mid z, s)\|_2^2 = \mathbb{E}_{t,q(z),p_t(a_t|z,s)}\|v_t(a_t \mid s) - (a_1 - a_0)\|_2^2,$$

where $q(z)$ is a coupling over the states—e.g., independent coupling $q(z) = p_0(a_0)p_1(a_1)$—and in the last equality we substitute $v^*(a_t \mid z, s) = a_1 - a_0$ with its analytic linear speed target velocity.

3 Flow Map Q-Guidance

Generative policies operate over an inner-time axis, i.e., ODE simulation time, that is distinct from the evolution of the MDP. Consequently, for every state along the trajectory s^τ, a corresponding action a^τ_1 must be generated through numerical simulation of the flow-policy ODE in eq. (1)—necessitating large amounts of function evaluations of the policy network. We next address this computational inefficiency through learning the flow-map, which dramatically speeds up simulation by taking large jumps along the ODE trajectory. We organize the remainder of this section as follows: in §3.1 we introduce *flow-map policies* and apply them to offline RL in §3.2. In §3.3 we rigorously design an efficient online adaptation update using a trust-region based on the critic's Q-function. Finally, in §3.4 we introduce our stochastic sampling approach to refine generated actions at inference.

3.1 Flow Map Policies

For high-fidelity action generation using flow-matching policies, it remains critical to simulate the infinitesimal dynamics of the parametrized velocity field in eq. (1). Instead of solving the ODE, we can parametrize and learn the unique two-time operator associated with the flow-matching policy.

Definition 3.1 (Flow-Map Policy). *Let $X_{r,t} : [0, 1]^2 \times \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}^d$ be a flow map that evolves the action dynamics between any $(r, t) \in [0, 1]$, conditioned on the MDP state $s \in \mathcal{S}$ governed*

by eq. (1), and satisfying the jump condition $X_{r,t}(a_r|s) = a_t$. The flow-map policy is then the distribution induced by this map evaluated at time $t = 1$, where $\pi(a|s) = [X_{r,1}]_\# p_r(a_r|s)$.

To parametrize the underlying flow map that induces this policy, we leverage the *average action velocity* $u_{r,t} : [0,1]^2 \times \mathcal{S} \times \mathbb{R}^d \to \mathbb{R}^d$, between the two time points r, t with the condition $r \leq t$:

$$X_{r,t}(a_r \mid s) = a_r + (t-r)u_{r,t}(a_r \mid s), \quad u_{r,t}(a_r \mid s) = \frac{1}{t-r}\int_r^t v_\tau(a_\tau \mid s)d\tau. \tag{2}$$

We note that, using eq. (2), we take jumps of size $t-r$ along the ODE trajectory. Furthermore, evaluating this flow map at the boundaries $r = 0$ and $t = 1$ yields the one-step policy $X_{0,1}$. We also highlight that the instantaneous velocity corresponds to the flow-matching policy can be recovered by taking the time limit yielding the tangent condition: $\lim_{r\to t}\partial_t X_{r,t}(a_r \mid s) = u_{t,t}(a_t|s) := v_t(a_t \mid s)$. As a result, this allows a supervision signal along the time diagonal $r = t$ amounting to classical flow-matching,

$$\mathcal{L}_{\text{Diag}} = \mathbb{E}_{t,q(z),p_t(a_t|z,s)}\|u_{t,t}(a_t \mid s) - (a_1 - a_0)\|_2^2. \tag{3}$$

To train the underlying flow map on the off-diagonal $r < t$, we follow the standard practice of enforcing consistency rules that are derived from satisfying the flow-map jump condition, as well as the semi-group property of the ODE [Boffi et al., 2025]. This leads to PINN style ℓ_2-regression objectives that *distill* the approximated ODE velocity field into $X_{r,t}$ by enforcing the Lagrange, Euler, and Progressive conditions of the flow map on the off-diagonal times $r < t$ combined with a stop-gradient $\texttt{sg}$:

1. **Lagrangian policy-distillation**.

$$\mathcal{L}_{\text{LPD}} = \int_0^1\int_0^t \mathbb{E}_{p_r(a_r|z,s)}\left[|\partial_t X_{r,t}(a_r \mid s) - \texttt{sg}\left(u_{t,t}(X_{r,t}(a_r \mid s))\right)|^2\right] drdt, \tag{4}$$

2. **Eulerian policy-distillation**.

$$\mathcal{L}_{\text{EPD}} = \int_0^1\int_0^t \mathbb{E}_{p_r(a_r|z,s)}\left[|\partial_r X_{r,t}(a_r \mid s) + \texttt{sg}\left(\nabla X_{r,t}(a_r \mid s)u_{r,r}(a_r \mid s)\right)|^2\right] drdt, \tag{5}$$

3. **Progressive policy-distillation**.

$$\mathcal{L}_{\text{PPD}} = \int_0^1\int_0^t \mathbb{E}_{p_r(a_r|z,s)}\left[|X_{w,t}(X_{r,w}(a_r \mid s)) - \texttt{sg}\left(X_{r,t}(a_r \mid s)\right)|^2\right] drdtd\gamma, \tag{6}$$

   where $w = (1-\gamma)r + \gamma t$ with $\gamma \in [0,1]$.

Relation to mean-flow policies. Critically, in contrast to prior work, equating policy learning with flow-maps unlocks the entire arsenal of flow-map-based learning objectives—with mean flows policies [Geng et al., 2025, Nguyen and Yoo, 2026, Zhan et al., 2026] being a specific instantiation of the Eulerian policy. In particular, mean-flow policies can be derived as a specific instance of the Eulerian policy distillation objective outlined in eq. (5) above (see §A.2), and the instantaneous velocity constraint is an application of the tangent condition and is simply the diagonal loss in eq. (3).

3.2 Offline RL with Flow Map Policies

We now deploy flow-map policies for offline-to-online reinforcement learning within an actor-critic framework. We first pre-train an efficient flow-map actor on an existing offline dataset $\mathcal{D} = \{(s, a_1, r, s')\}$, along with a critic Q-network. We parametrize the actor as $u_{r,t}(a_r|s)$ over all time pairs $(r,t) \in [0,1]^2$, trained with the policy self-distillation objectives from §3.1: $\mathcal{L}_{\text{actor}}^{\text{off}} = \mathcal{L}_{\text{Diag}} + \lambda\,\mathcal{L}_{\text{SD}}$, where $\mathcal{L}_{\text{SD}}$ corresponds to any of the policy self-distillation losses and λ is a hyper-parameter that controls the strength of off-diagonal training. For maximally efficient action generation, we can simply invoke the flow-map policy $a_1 = X_{0,1}^{\text{off}}(a_0|s)$ and generate actions in a single forward pass by directly transporting the prior noisy action a_0 to the clean action a_1 in one step:

$$a_1 = a_0 + u_{0,1}^{\text{off}}(a_0 \mid s), \quad a_0 \sim \mathcal{N}(0, I). \tag{7}$$

We train the critics via clipped double Q-learning with EMA targets Q_{ϕ_j} [Fujimoto et al., 2018]:

$$\mathcal{L}_{\text{critic}}(\phi_j) = \mathbb{E}_{(s,a_1,r,s')\sim\mathcal{D}}\left[\left(Q_{\phi_j}(s,a_1) - y\right)^2\right], \quad y = r + \gamma \min_{j=1,2} Q_{\phi_j}(s', X_{0,1}^{\text{off}}(a_0' \mid s')). \tag{8}$$

3.3 Efficient Trust-Region Based Online Adaptation

Transitioning to the online phase introduces the challenge of identifying the optimal action to imitate when training generative policies: in continuous action spaces without a curated dataset, solving $\arg\max_a Q(s,a)$ as a learning target is intractable. A common strategy is the "best-of-N" heuristic, which draws N actions from the policy and selects the one with the highest Q-value [Zhan et al., 2026]. However, this naive strategy imposes a non-trivial drawback of requiring a large number of sampled actions N, which requires at minimum N one-step simulations of the flow-map actor and Q-function evaluations. We next develop a more principled approach that finds the optimal action by constructing a trust region. Consider a flow-map policy $\pi^{\text{off}}(\cdot|s)$, the natural question for online adaptation is:

Q. *What is the optimal perturbation Δ for $a_1 \sim \pi^{off}$ that maximizes the critic's Q-function?*

To answer this question, we assume the existence of an optimal action $a_1^* = a_1 + \Delta^*$ that is feasible—i.e., reachable from the flow-map policy via a perturbation Δ. To prevent unbounded deviation from $a_1 \sim \pi^{\text{off}}(\cdot|s)$, we constrain Δ within a trust region of radius η around the critic's current Q-value. This yields the following non-linear optimization problem that maximizes the critic's Q-function:

$$\arg\max_{\Delta} \quad \mathbb{E}_{r\sim\mathcal{U}[0,1)}\left[Q_\phi\big(s, X^{\text{off}}_{0,r}(a_0 \mid s) + \Delta\big)\right] \quad \text{s.t.} \qquad \|\Delta\|_2 \le \eta \tag{9}$$

In the case where the Δ-perturbation is given as the average velocity network $u_{r,1}(a_r|s)$, constraining the perturbation $\|\Delta\|_2 \le \eta$ in action space is equivalent to bounding $\|u_{r,1}(a_r|s) - u^{\text{off}}_{r,1}(a_r|s)\|_2$. As the critic Q-function is non-linear, this optimization problem is challenging to solve in closed form. Instead, we can consider a first-order approximation of optimality that aims to find *optimal target displacement* to any generic reference $u^{\text{ref}}_{r,1}(a_r|s)$. Interestingly, under these settings, the analytic expression of the optimal target displacement admits a closed-form expression.

Theorem 3.2. *Consider a flow-map policy $\pi^{ref}(\cdot|s)$ with underlying flow map $X^{ref}_{r,1}$, generating actions $a_1 = a_r + (1-r)\, u^{ref}_{r,1}(a_r \mid s)$. The optimal average velocity $u^*_{r,1}$ that maximizes the first-order expansion of Q_ϕ around a_1, subject to trust-region constraint $\|u_{r,1} - u^{ref}_{r,1}\|_2 \le \eta$, is:*

$$u^*_{r,1}(a_r \mid s) = u^{ref}_{r,1}(a_r \mid s) + \eta\, \frac{\nabla_a Q_\phi(s,a_1)}{\|\nabla_a Q_\phi(s,a_1)\|_2}. \tag{10}$$

Proof Sketch. To maximize Q_ϕ over $u_{r,1}$, we substitute the flow-map parameterization into a first-order Taylor expansion around a_1, which reduces the problem to maximizing $\langle \nabla_a Q_\phi(s,a_1),\, u_{r,1} - u^{\text{ref}}_{r,1}\rangle$ subject to $\|u_{r,1} - u^{\text{ref}}_{r,1}\|_2 \le \eta$. Solving the associated KKT conditions yields the optimal closed-form solution. The full proof is provided in §A.1. □

Theorem 3.2 holds for any reference flow-map policy. Setting $\pi^{\text{ref}} = \pi^{\text{off}}$, i.e., anchoring to the offline flow-map velocity $u^{\text{off}}_{r,1}(a_r|s)$, results in the following optimal average velocity:

$$u^*_{r,1}(a_r|s) = u^{\text{off}}_{r,1}(a_r|s) + \eta\, \frac{\nabla_a Q_\phi(s,a_1)}{\|\nabla_a Q_\phi(s,a_1)\|_2}. \tag{11}$$

The analytic form of eq. (11) enables us to form a learning target for efficient online adaptation that we term FLOW MAP Q-GUIDANCE (FMQ). Specifically, we construct the interpolant $a_r = (1-r)\, a_0 + r\, a_{\text{data}}$ using noise $a_0 \sim \mathcal{N}(0, I)$ and actions from a replay buffer $a_{\text{data}} \sim \mathcal{D}$. This allows to then regress $u_{r,1}(a_r|s)$ against the optimal self-bootstrapped trust-region target below:

$$\mathcal{L}_{\text{FMQ}}(\theta) = \mathbb{E}_{r,\, a_0,\, a_{\text{data}}}\left[\left\| u^{\theta}_{r,1}(a_r) - \text{sg}\left(u^{\text{off}}_{r,1}(a_r) + \eta \frac{\nabla_a Q_\phi(s,a_1)}{\|\nabla_a Q_\phi(s,a_1)\|_2 + \kappa_1} \right) \right\|_2^2\right], \tag{12}$$

where $\text{sg}(\cdot)$ is the stop-gradient operator, and $\kappa_1 > 0$ is a stability constant as described in algorithm 1.

Uncertainty-Aware Adaptive Trust Region. A fixed radius η applies the same step size regardless of critic's confidence. We now formulate an adaptive per-sample η-radius driven by a cheap heuristic, driven by capturing the epistemic uncertainty in the critic ensemble. Given a twin-critic ensemble, we define $\delta_{\text{critic}}(s,a) = \frac{1}{\sqrt{2}}|Q_{\phi_1}(s,a) - Q_{\phi_2}(s,a)|$ that captures the absolute discrepancy of Q-values amongst the critics. This allows us to design a batch-normalized per-sample *effective* trust region,

$$\eta_{\text{eff}}(s,a) = \frac{1}{1 + \beta\, \tilde{\delta}_{\text{critic}}(s,a)}, \qquad \tilde{\delta}_{\text{critic}}(s,a) = \frac{\delta_{\text{critic}}(s,a)}{\frac{1}{B}\sum_{i=1}^{B} \delta_{\text{critic}}(s_i,a_i) + \kappa_2}, \tag{13}$$

where $\kappa_2 > 0$ is a small constant, and β a hyper-parameter. By construction, $\eta_{\text{eff}} \in (0, 1]$ decays monotonically with the magnitude $\delta_{\text{critic}}(s, a)$: a small discrepancy δ_{critic} leads to larger steps, while conversely a larger discrepancy δ_{critic} results in the prioritization of the offline flow map actor.

3.4 Inference-Time Q-Guided Search

The flow map policy induces a mapping that transports noisy actions marginals $[X_{r,t}]_{\#}p_r = p_t$ for all $r, t \in [0, 1]$. This mapping is fundamentally incapable of capturing the conditional posterior over endpoints that also maximize a critic's Q-value. As a result, the initial sampling of a_0 may have a disproportionate impact on the solutions to the optimization problem in eq. (9). Instead of training a separate stochastic flow map for reward alignment [Potaptchik et al., 2026, Holderrieth et al., 2026] we opt for a purely inference-time search strategy. Specifically, we next construct a stochastic sampler for flow map actors that also leverages the trust region of the critic's Q-value.

Stochastic Sampling with SNR. To design a stochastic sampler, we leverage a renoising strategy based on the signal-to-noise ratio (SNR). In particular, given the one-step flow map actor *after online adaptation* $a_1 = X^{\text{adapt}}_{0,1}(a_0|s)$, we can re-noise by judiciously selecting a new time $t' < 1$. To do so, we design the re-noising interpolant by selecting $t' = \text{SNR}/(1+\text{SNR}) \in (0, 1)$:

$$a_{t'} = t' \cdot a_1 + (1 - t') \cdot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I), \tag{14}$$

A second application of the flow map then transports this intermediate state back to time $t = 1$, i.e., $\tilde{a}_1 = X^{\text{adapt}}_{t',1}(a_{t'} \mid s)$. Crucially, each draw of ε yields a *different* action $\tilde{a}_1$—a stochastic sample from the flow map to which the trust-region update can be re-applied. The approach thus defines an iterative refinement procedure during inference: (1) we corrupt the current actions a_1 to t' using eq. (14), gaining access to diverse actions noisy intermediate states a'_t, and (2) then re-apply the optimal trust-region Q-value projection to each new sample, obtaining $\tilde{a}_1$. The noise level t' controls the exploration–exploitation balance: small t' (low SNR) places $a_{t'}$ closer to pure noise, allowing the flow map to explore distant modes. Conversely, a large t' (high SNR) preserves most of the current action a_1, it restricts the update to a more local refinement.

Q-Guided Beam Search. We now outline an inference-time search strategy that combines the stochastic sampler with beam search. This new algorithm is deployed only once at inference — i.e., inference-time scaling via search — and, as a result, does not affect training speed for online updates. We provide the full algorithmic description in algorithm 2. We instantiate this new final inference procedure Q-GUIDED BEAM SEARCH (QGBS), which balances exploration and exploitation. Specifically, QGBS operates over M particles that are refined over K steps along the already online-adapted flow map policy trajectory. In summary, the algorithm follows the following two steps iteratively:

1. *Exploration:* The first step in QGBS constitutes an exploration phase that diversifies the candidate N actions to a batch B of intermediate states that then yields a total of $\bigcup_{i=1}^{N \cdot B} X^{\text{adapt}}_{t',1}(a'_t|s)$ actions.
2. *Exploitation:* The second step selects the most promising M particles using the critic $Q_\phi(s, \tilde{a}_1)$, which are then used in the trust region update (eq. (11)), before progressing to the next beam.

After K steps, the procedure returns $\arg\max_i Q_\phi(s, a_i)$. When K=0, the method reduces to best-of-M Q-Steering (a single application of Theorem 3.2 without iteration).

4 Experiments

We investigate the application of FMQ across 12 robotic manipulation and locomotion tasks with varying difficulties across 7 environments from the OGBench [Park et al., 2024] and Robomimic [Mandlekar et al., 2021] benchmarks. The manipulation tasks include two from Robomimic (`can`, `square`) and six from OGBench (`cube-dbl-t3/4`, `cube-trl-t3/4`, `scene-t4/5`). Locomotion is evaluated on four OGBench tasks (`hmaze-med-t3/4`, `amaze-gnt-t4/5`). During offline pre-training, we use multi-human demonstration datasets for Robomimic and the default noisy-expert datasets (play-style and navigate) for OGBench. In addition, we utilize single-task OGBench variants for offline-to-online RL. The humanoid and ant maze tasks use sparse rewards, while all others use dense rewards. For clarity, we report full training configurations and experimental setups in §H.

Baselines. We compare FMQ against two main baselines: (1) QC [Li et al., 2025] trains a multi-step flow matching policy with 10 integration steps. (2) As our second baseline, we report the state-of-the-

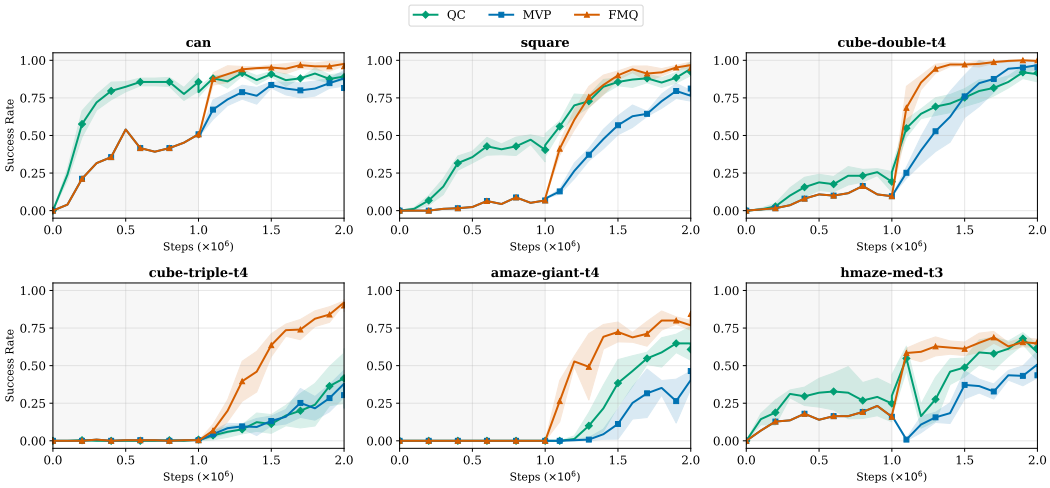

Figure 2: Training curves on 6 environments. Average success rate at every 100K during 1M offline followed by 1M online steps over 5 seeds. Shaded regions indicate 95% CIs.

art method MVP [Zhan et al., 2026], which trains a mean flow policy with an initial velocity constraint. For MVP, we distinguish MVP* as results taken directly from the original paper, which is only available in $6/12$ environments considered here, from our reproduction MVP, allowing us to investigate all considered environments. All baselines share the same model architecture and follow the same clipped double Q-learning algorithm [Fujimoto et al., 2018]. At inference time, QC, MVP, and our method FMQ all select actions via best-of-32 sampling, choosing the action with the highest Q-value.

Evaluation protocol. We follow the standard offline-to-online protocol from Park et al. [2024]: 1M gradient steps of offline pre-training using the provided dataset, followed by 1M steps of online fine-tuning with environment interaction. During the online phase, newly collected transitions are appended to the replay buffer. To monitor training, we evaluate the policy every 100K steps over 50 episodes with randomized initial states. Finally, we evaluate the last checkpoint across 50 unseen test episodes per environment and compute the average success rate across 5 seeds and report the Interquartile Mean (IQM) alongside 95% stratified bootstrap confidence intervals [Agarwal et al., 2021].

4.1 Main Results

We report our main quantitative results in table 1 and observe that FMQ achieves the highest IQM score $(0.91; [0.89, 0.93])$, outperforms MVP by 21.3% $(0.75; [0.73, 0.77])$ and QC by 5.8% $(0.86; [0.84, 0.87])$. The improvement is most pronounced on challenging environments: on `cube-trl-t4`, FMQ reaches 0.88 compared to 0.37 for QC and 0.32 for MVP, and on `amaze-gnt-t4`, FMQ achieves 0.80 compared to 0.64 and 0.42, respectively. We also note that QC outperforms MVP on average $(0.76$ vs. $0.68)$, but this comes at $10\times$ computational overhead at inference due to simulation of the flow rather than one-step generation. Nevertheless, we find that FMQ outperforms QC in $10/12$ environments using only a single generation step. These results demonstrate the benefit of leveraging the optimal Q-guidance in FMQ in comparison to best-of-N.

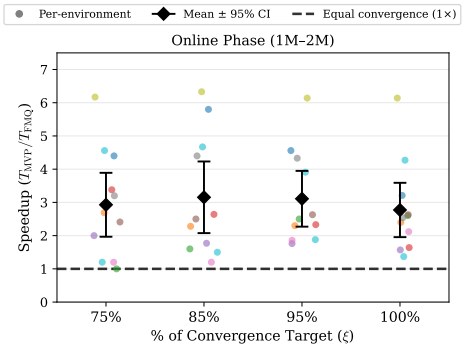

Figure 3: Convergence speedup of FMQ compared to MVP at success targets (ξ), with 95% CIs.

Inference scaling. To evaluate our complementary contribution Q-Guided Beam Search (QGBS) that can be used as a stochastic sampler on any flow-map policy, including the baseline method MVP. We introduce two additional configurations, MVP + QGBS and FMQ + QGBS. Specifically, we replace the best-of-N sampling in inference time with our stochastic sampling algorithm, which is combined with beam search ($K = 1$, $B = 4$, $M = 4$) and outlined in §3.4. Overall, applying QGBS we observe a relative increase in IQM by 8.0% (from 0.75 to 0.81)

Table 1: Success rate (mean $\pm$ std over 5 seeds, 50 episodes). Best per row in **bold**, second best underlined. Aggregate performance is measured by the IQM scores with 95% CIs.

Environment	QC	MVP*	MVP	MVP + QGBS (Ours)	FMQ (Ours)	FMQ + QGBS (Ours)
`can`	0.88 ± 0.06	0.92 ± 0.07	0.83 ± 0.07	0.87 ± 0.07	0.96 ± 0.04	$\mathbf{0.97 \pm 0.03}$
`square`	0.89 ± 0.04	0.93 ± 0.01	0.82 ± 0.04	0.83 ± 0.05	0.94 ± 0.02	$\mathbf{0.95 \pm 0.04}$
`cube-dbl-t3`	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
`cube-dbl-t4`	0.92 ± 0.05	0.95 ± 0.04	0.98 ± 0.02	0.98 ± 0.02	0.98 ± 0.02	$\mathbf{1.00 \pm 0.00}$
`cube-trl-t3`	0.83 ± 0.08	0.71 ± 0.06	0.64 ± 0.12	0.78 ± 0.12	0.78 ± 0.10	$\mathbf{0.84 \pm 0.04}$
`cube-trl-t4`	0.37 ± 0.26	0.52 ± 0.11	0.32 ± 0.07	0.37 ± 0.09	$\mathbf{0.88 \pm 0.07}$	0.87 ± 0.05
`scene-t4`	0.99 ± 0.01	—	0.92 ± 0.02	0.98 ± 0.02	$\mathbf{1.00 \pm 0.00}$	0.99 ± 0.01
`scene-t5`	0.96 ± 0.02	—	0.90 ± 0.06	0.95 ± 0.05	0.98 ± 0.02	$\mathbf{1.00 \pm 0.00}$
`hmaze-med-t3`	0.65 ± 0.11	—	0.47 ± 0.10	0.53 ± 0.03	$\mathbf{0.69 \pm 0.04}$	0.58 ± 0.07
`hmaze-med-t4`	0.04 ± 0.03	—	0.00 ± 0.00	0.02 ± 0.02	$\mathbf{0.06 \pm 0.03}$	$\mathbf{0.06 \pm 0.03}$
`amaze-gnt-t4`	0.64 ± 0.12	—	0.42 ± 0.06	0.43 ± 0.04	$\mathbf{0.80 \pm 0.06}$	0.77 ± 0.03
`amaze-gnt-t5`	0.91 ± 0.05	—	0.82 ± 0.08	0.90 ± 0.06	$\mathbf{0.92 \pm 0.04}$	$\mathbf{0.92 \pm 0.05}$
IQM SR [95% CI]	0.86 $[0.84, 0.87]$	—	0.75 $[0.73, 0.77]$	0.81 $[0.78, 0.83]$	0.91 $[0.89, 0.93]$	**0.93** $[\mathbf{0.91}, \mathbf{0.94}]$

for MVP and a relative increase by 2.2% (from 0.91 to 0.93) for FMQ. However, for a small number of locomotion tasks (`hmaze-med-t3` and `amaze-gnt-t4`), QGBS degrades performance by 15.9% and 3.8% respectively. Overall, we find that combining both our proposed training and inference algorithms FMQ + QGBS leads to the best performance, achieving the highest IQM of 0.93; $[0.91, 0.94]$, with non-overlapping confidence intervals against all baselines, including QC (0.86; $[0.84, 0.87]$) and MVP (0.75; $[0.73, 0.77]$). These results highlight the impact of performance by increasing the compute budget at inference through stochastic sampling and beam search.

4.2 Sample Efficiency Analysis

We next investigate the sample efficiency gains of using Q-guidance to train our flow map policies. In fig. 2, we plot the training curves during the online adaptation for all methods across 6 environments (see fig. 17 for full). We find that FMQ consistently converges faster than MVP during online fine-tuning, despite both methods starting from the same offline checkpoint. To quantify this advantage, we define ξ as the highest success rate that MVP and FMQ can reach, computed per seed and environment. In table 4, we measure speedup T: the number of steps to first reach $\{75\%, 85\%, 95\%, 100\%\}$ of ξ. The speedup ratio $T_{\text{MVP}}/T_{\text{FMQ}}$, averaged over 5 seeds, quantifies how many times faster FMQ converges to each fraction of ξ. In the online phase (1M–2M), FMQ reaches the highest success rate achievable by MVP $2.77\times$ faster on average, and up to $6.14\times$ on `hmaze-med-t3`. These results further confirm that Q-gradient alignment provides a stronger learning signal than best-of-N selection, leading to faster policy improvement per environment step.

4.3 Ablation Studies

Table 2: QGBS efficiency ablation.

K	$\{B, M\}$	NFE	IQM
0	$\{1, 32\}$	32	0.91 $[0.89, 0.93]$
1	$\{4, 4\}$	20	**0.93** $[\mathbf{0.91}, \mathbf{0.94}]$
1	$\{2, 8\}$	24	**0.93** $[\mathbf{0.91}, \mathbf{0.95}]$
1	$\{1, 16\}$	32	0.92 $[0.90, 0.93]$
1	$\{4, 16\}$	80	0.92 $[0.90, 0.93]$
2	$\{4, 4\}$	36	0.91 $[0.90, 0.93]$
2	$\{1, 16\}$	48	0.90 $[0.89, 0.92]$
2	$\{4, 16\}$	144	0.90 $[0.89, 0.91]$

Inference-time Beam Search. The computational cost of utilizing QGBS is NFE $= M(1 + KB)$ per action selection, where M is the number of initial candidates, K is the number of re-noising steps, and B is the number of completions per candidate. We note that best-of-N sampling corresponds to K=0 and M=N. In table 2, we show that the optimal configuration (K=1, B=4, M=4) achieves a peak IQM of 0.93 with only 20 FE—37.5% fewer than best-of-32—suggesting that diversifying candidates through renoising is more efficient than considering more candidates. Increasing K beyond 1 does not improve performance, suggesting that only a modest increase in inference cost is needed for optimal performance.

Trust-Region Convergence Analysis. We evaluate the impact of trust-region in eq. (12). Specifically, we measure the distance $\|u^{\text{on}}_{r,1} - u^{\text{off}}_{r,1}\|_2$ relative to the frozen offline policy. At the onset of online training, the online and offline velocity fields coincide, and the distance is 0. As online training begins, the learning actively drives $u^{\text{on}}_{r,1}$ toward $u^{\text{off}}_{r,1} + \eta \nabla_a Q_\phi(s, a_1)/\|\nabla_a Q_\phi(s, a_1)\|_2$—as evidenced in fig. 4—and stabilizes near η_{eff}. Thus, the flow map policy incorporates the normalized Q-gradient direction while remaining safely constrained within the trust-region radius (c.f. fig. 18 for all environments).

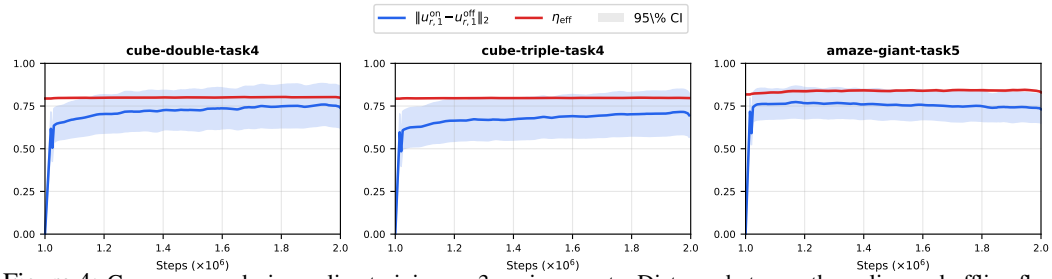

Figure 4: Convergence during online training on 3 environments. Distance between the online and offline flow map policies (blue) converges to the adaptive radius η_{eff} (red) as the policy incorporates the Q-guidance.

Flow map policy variants. We next ablate the offline flow map policy variants and their impact on performance. In table 3. ESD and LSD achieve the same IQM of 0.79, while PSD lags slightly at 0.77. However, ESD exhibits the narrowest 95% bootstrap CI $[0.77, 0.81]$, indicating it is the most consistent across environments. We therefore adopt ESD as the default formulation for our presented results.

5 Related work

Generative Policies. Diffusion models and flow matching have emerged as expressive policy representations [Chi et al., 2023, Pearce et al., 2023]. For policy learning, prior methods train diffusion and flow-matching models via weighted behavioral cloning [Lu et al., 2023, Kang et al., 2023], reparameterized policy gradients [Wang et al., 2023, Ding and Jin, 2023, Zhang et al., 2024], and rejection sampling [Chen et al., 2024, Hansen-Estruch et al., 2023, He et al., 2024]. While effective, reparameterized gradients require costly backpropagation through time (BPTT). To address the latency of flow-matching multi-step models, FQL [Park et al., 2025] distills a multi-step flow into a separate one-step student network, while QC [Li et al., 2025] groups action sequences. MVP [Zhan et al., 2026] natively achieves one-step generation but uses a "generate-and-select" heuristic to find imitation targets. Concurrently, RFM [Li et al., 2026] seeks to target the unnormalized Q-distribution with posterior mean estimates during training. In contrast, we formalize one-step flow map policies and leverage FMQ for more efficient online adaptation via an optimal, closed-form Q-guided trust region.

Table 3: Self-distillation loss ablation.

Environment	ESD	PSD	LSD
`cube-trl-t3`	0.79 ± 0.06	0.74 ± 0.13	$\mathbf{0.84 \pm 0.09}$
`cube-trl-t4`	$\mathbf{0.90 \pm 0.05}$	0.87 ± 0.09	0.88 ± 0.06
`hmaze-med-t3`	$\mathbf{0.64 \pm 0.02}$	0.48 ± 0.29	0.62 ± 0.09
`hmaze-med-t4`	0.03 ± 0.03	0.04 ± 0.04	$\mathbf{0.06 \pm 0.00}$
`amaze-gnt-t4`	0.82 ± 0.02	$\mathbf{0.86 \pm 0.04}$	0.78 ± 0.06
`amaze-gnt-t5`	$\mathbf{0.92 \pm 0.01}$	0.87 ± 0.05	0.91 ± 0.03
IQM	**0.79** [0.77, 0.81]	0.77 [0.70, 0.81]	**0.79** [0.75, 0.82]

Offline-to-Online RL. Offline-to-online RL accelerates online learning by initializing from a static dataset [Levine et al., 2020]. However, offline RL must contend with overestimation of Q-values for out-of-distribution actions, addressed through divergence penalties [Fujimoto et al., 2019, Wu et al., 2019, Nair and Dalal, 2020, Wang et al., 2023], pessimistic value estimates [Kumar et al., 2020, Yu et al., 2020, An et al., 2021], or in-sample maximization [Kostrikov et al., 2022, Garg et al., 2023]. When transitioning to the online phase, distribution shift can cause catastrophic forgetting of the behavioral prior [Lee et al., 2022, Song et al., 2023, Nakamoto et al., 2023]. Recent state-of-the-art flow-matching methods rely on behavioral regularization [Park et al., 2025] or best-of-N selection [Li et al., 2025, Zhan et al., 2026] to stabilize adaptation. In contrast, we formulate online fine-tuning as a trust-region problem with FMQ.

6 Conclusion

In this paper, we bridge the gap between expressive generative policies and the low-latency requirements of offline-to-online RL. We formulate online adaptation of one-step flow map policies as a trust-region optimization problem, yielding FLOW MAP Q-GUIDANCE (FMQ): a closed-form, locally optimal Q-guided update that improves a linearized critic while remaining anchored to the offline behavioral prior. We further introduce QGBS, an inference-time refinement procedure based on stochastic renoising and beam search that improves any flow map policy. Across 12 continuous-control tasks from OGBench and RoboMimic, FMQ establishes state-of-the-art performance, outperforming the previous leading one-step policy, MVP, by a 21.3% relative margin in IQM success.

Acknowledgments

We thank Jorge de Freitas for their helpful comments on an earlier draft and Luca Eyring for the insightful discussion on renoising. This work was supported by UKRI (EP/Y037111/1) as part of the ProSafe project (EU Horizon 2020, MSCA, grant no. 101119358).

References

R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021. (Cited on page 7)

M. S. Albergo, N. M. Boffi, and E. Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions, 2023. URL https://arxiv.org/abs/2303.08797. (Cited on pages 1 and 3)

G. An, S. Moon, J.-H. Kim, and H. O. Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. *Advances in neural information processing systems*, 34:7436–7447, 2021. (Cited on page 9)

N. M. Boffi, M. S. Albergo, and E. Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation, 2025. URL https://arxiv.org/abs/2505.18825. (Cited on page 4)

T. Chen, Z. Wang, and M. Zhou. Diffusion policies creating a trust region for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 37:50098–50125, 2024. (Cited on page 9)

C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023. (Cited on page 9)

C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025. (Cited on page 1)

O.-E. Collaboration, A. O'Neill, A. Rehman, A. Gupta, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 1(2), 2023. (Cited on page 1)

Z. Ding and C. Jin. Consistency models as a rich and efficient policy class for reinforcement learning. *arXiv preprint arXiv:2309.16984*, 2023. (Cited on page 9)

J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020. (Cited on page 1)

S. Fujimoto and S. S. Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 20132–20145, 2021. (Cited on page 1)

S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, 2018. URL https://api.semanticscholar.org/CorpusID:3544558. (Cited on pages 4, 7, and 22)

S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062. PMLR, 2019. (Cited on page 9)

D. Garg, J. Hejna, M. Geist, and S. Ermon. Extreme q-learning: Maxent rl without entropy. In *International Conference on Learning Representations*, 2023. (Cited on page 9)

Z. Geng, M. Deng, X. Bai, J. Z. Kolter, and K. He. Mean flows for one-step generative modeling, 2025. URL https://arxiv.org/abs/2505.13447. (Cited on page 4)

P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine. Idql: Implicit q-learning as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023. (Cited on page 9)

L. He, L. Shen, and X. Wang. Aligniql: Policy alignment in implicit q-learning through constrained optimization. *arXiv preprint arXiv:2405.18187*, 2024. (Cited on page 9)

P. Holderrieth, D. Chen, L. Eyring, I. Shah, G. Anantharaman, Y. He, Z. Akata, T. Jaakkola, N. M. Boffi, and M. Simchowitz. Diamond maps: Efficient reward alignment via stochastic flow maps. *arXiv preprint arXiv:2602.05993*, 2026. (Cited on page 6)

N. Jaques, A. Ghandeharioun, J. H. Shen, C. Ferguson, A. Lapedriza, N. Jones, S. Gu, and R. Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*, 2019. (Cited on page 1)

G. Kahn, A. Villaflor, P. Abbeel, and S. Levine. Composable action-conditioned predictors: Flexible off-policy learning for robot navigation. In *Conference on robot learning*, pages 806–816. PMLR, 2018. (Cited on page 1)

B. Kang, X. Ma, C. Du, T. Pang, and S. Yan. Efficient diffusion policies for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:67195–67212, 2023. (Cited on page 9)

I. Kostrikov, A. Nair, and S. Levine. Offline reinforcement learning with implicit q-learning. In *International Conference on Learning Representations*, 2022. (Cited on page 9)

A. Kumar. Data-driven deep reinforcement learning. *Berkeley Artificial Intelligence Research (BAIR), Tech. Rep*, 2019. (Cited on page 1)

A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191, 2020. (Cited on page 9)

J. Lee, C. Paduraru, D. J. Mankowitz, N. Heess, D. Precup, K.-E. Kim, and A. Guez. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, pages 1602–1612. PMLR, 2022. (Cited on page 9)

S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. (Cited on pages 1 and 9)

Q. Li, Z. Zhou, and S. Levine. Reinforcement learning with action chunking. *arXiv preprint arXiv:2507.07969*, 2025. (Cited on pages 6, 9, and 22)

Z. Li, S. Tang, and N. Azizan. Reverse flow matching: A unified framework for online reinforcement learning with diffusion and flow policies. In *International Conference on Machine Learning*, 2026. (Cited on page 9)

Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. (Cited on pages 1 and 3)

X. Liu, C. Gong, and Q. Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022. (Cited on pages 1 and 3)

C. Lu, Y. Hu, et al. Contrastive energy prediction for exact energy-guided diffusion sampling in offline reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2023. (Cited on page 9)

A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021. (Cited on page 6)

A. Nair and M. r. l. w. o. d. Dalal. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020. (Cited on page 9)

M. Nakamoto, Y. Zhai, A. Singh, M. Radin, A. Kumar, C. Finn, and S. Levine. Cal-ql: Calibrated offline rl pre-training for efficient online fine-tuning. In *Advances in Neural Information Processing Systems*, volume 36, 2023. (Cited on page 9)

T. X. Nguyen and C. D. Yoo. One-step flow q-learning: Addressing the diffusion policy bottleneck in offline reinforcement learning. In *The Fourteenth International Conference on Learning Representations*, 2026. (Cited on page 4)

S. Park, K. Frans, B. Eysenbach, and S. Levine. Ogbench: Benchmarking offline goal-conditioned rl. *arXiv preprint arXiv:2410.20092*, 2024. (Cited on pages 6 and 7)

S. Park, Q. Li, and S. Levine. Flow q-learning. In *International Conference on Machine Learning (ICML)*, 2025. (Cited on pages 2 and 9)

T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, and S. Devlin. Imitating human behaviour with diffusion models. *ArXiv*, abs/2301.10677, 2023. URL https://api.semanticscholar.org/CorpusID:256231177. (Cited on page 9)

S. Peluchetti. Non-denoising forward-time diffusions. *arXiv preprint arXiv:2312.14589*, 2023. (Cited on page 3)

P. Potaptchik, A. Saravanan, A. Mammadov, A. Prat, M. S. Albergo, and Y. W. Teh. Meta flow maps enable scalable reward alignment. *arXiv preprint arXiv:2601.14430*, 2026. (Cited on page 6)

J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015. (Cited on page 1)

Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020. (Cited on page 1)

Y. Song, Y. Zhou, A. Sekhari, J. A. Bagnell, A. Krishnamurthy, and W. Sun. Hybrid rl: Using both offline and online data can make rl efficient. In *International Conference on Learning Representations*, 2023. (Cited on page 9)

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998. (Cited on page 2)

D. Tarasov, V. Kurenkov, A. Nikulin, and S. Kolesnikov. Revisiting the minimalist approach to offline reinforcement learning. *Advances in Neural Information Processing Systems*, 36:11592–11620, 2023. (Cited on page 1)

A. Tong, K. Fatras, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, G. Wolf, and Y. Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023. (Cited on page 3)

Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022. (Cited on page 1)

Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations*, 2023. (Cited on page 9)

Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019. (Cited on page 9)

L. Yang, Z. Huang, F. Lei, Y. Zhong, Y. Yang, C. Fang, S. Wen, B. Zhou, and Z. Lin. Policy representation via diffusion probability model for reinforcement learning. *arXiv preprint arXiv:2305.13122*, 2023. (Cited on page 2)

T. Yu, G. Thomas, L. Yu, S. Ermon, J. Zou, S. Levine, C. Finn, and T. Ma. Mopo: Model-based offline policy optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 14129–14142, 2020. (Cited on page 9)

G. Zhan, Y. Jiang, S. E. Li, Y. Lyu, X. Zhang, and Y. Yin. A transformation-aggregation framework for state representation of autonomous driving systems. *IEEE Transactions on Intelligent Transportation Systems*, 25(7):7311–7322, 2024. (Cited on page 2)

G. Zhan, X. An, Y. Jiang, J. Duan, H. Zhao, and S. E. Li. Physics informed neural pose estimation for real-time shape reconstruction of soft continuum robots. *IEEE Robotics and Automation Letters*, 2025. (Cited on page 2)

G. Zhan, L. Tao, P. Wang, Y. Wang, Y. Li, Y. Chen, H. Li, M. Tomizuka, and S. E. Li. Mean flow policy with instantaneous velocity constraint for one-step action generation. *arXiv preprint arXiv:2602.13810*, 2026. (Cited on pages 2, 4, 5, 7, 9, 15, and 23)

R. Zhang, Z. Luo, J. Sjölund, T. B. Schön, and P. Mattsson. Entropy-regularized diffusion policy with q-ensembles for offline reinforcement learning. *Advances in neural information processing systems*, 37:98871–98897, 2024. (Cited on page 9)

Z. Zhu, H. Zhao, H. He, Y. Zhong, S. Zhang, H. Guo, T. Chen, and W. Zhang. Diffusion models for reinforcement learning: A survey. *arXiv preprint arXiv:2311.01223*, 2023. (Cited on page 1)

A Theoretical details

A.1 Proofs

Theorem 3.2. *Consider a flow-map policy $\pi^{ref}(\cdot|s)$ with underlying flow map $X^{ref}_{r,1}$, generating actions $a_1 = a_r + (1-r)\, u^{ref}_{r,1}(a_r \mid s)$. The optimal average velocity $u^*_{r,1}$ that maximizes the first-order expansion of Q_ϕ around a_1, subject to trust-region constraint $\|u_{r,1} - u^{ref}_{r,1}\|_2 \le \eta$, is:*

$$u^*_{r,1}(a_r \mid s) = u^{ref}_{r,1}(a_r \mid s) + \eta\, \frac{\nabla_a Q_\phi(s, a_1)}{\|\nabla_a Q_\phi(s, a_1)\|_2}. \tag{10}$$

Proof. We seek the optimal average velocity $u_{r,1}(a_r \mid s)$ that generates action

$$\bar{a}_1 = a_r + (1-r)\, u_{r,1}(a_r \mid s). \tag{15}$$

Let $u_{r,1}(a_r \mid s)$ denote a candidate average velocity, generating action $\bar{a}_1 = a_r + (1-r)u_{r,1}(a_r \mid s)$. We take the first-order Taylor expansion of the Q-function around the reference action a_1:

$$Q_\phi(s, \bar{a}_1) \approx Q_\phi(s, a_1) + \langle \nabla_a Q_\phi(s,\, a_1),\, \bar{a}_1 - a_1 \rangle \tag{16}$$

Substituting the flow map parameterization, the starting state a_r cancels out, leading to the difference in average velocity vectors:

$$\bar{a}_1 - a_1 = \big(a_r + (1-r)u_{r,1}(a_r \mid s)\big) - \big(a_r + (1-r)u^{\text{ref}}_{r,1}(a_r \mid s)\big) = (1-r)(u_{r,1}(a_r \mid s) - u^{\text{ref}}_{r,1}(a_r \mid s)) \tag{17}$$

Since $Q_\phi(s, a_1)$ is constant with respect to $u_{r,1}(a_r \mid s)$, maximizing the linear approximation subject to the trust-region constraint on the average velocity is formulated as:

$$\begin{aligned} \min_{u_{r,1}} \quad & f_0(u_{r,1}(a_r \mid s)) = -\langle \nabla_a Q_\phi(s,\, a_1),\, u_{r,1}(a_r \mid s) - u^{\text{ref}}_{r,1}(a_r \mid s) \rangle \\ \text{subject to} \quad & f_1(u_{r,1}(a_r \mid s)) = \tfrac{1}{2}\|u_{r,1}(a_r \mid s) - u^{\text{ref}}_{r,1}(a_r \mid s)\|_2^2 - \tfrac{1}{2}\eta^2 \le 0 \end{aligned} \tag{18}$$

Because $\eta > 0$, the interior of the feasible set is non-empty, satisfying Slater's constraint qualification. Therefore, strong duality holds and the KKT conditions are necessary and sufficient. The Lagrangian is:

$$\begin{aligned} \mathcal{L}(u_{r,1}, \lambda) = & -\langle \nabla_a Q_\phi(s,\, a_1),\, u_{r,1}(a_r \mid s) - u^{\text{ref}}_{r,1}(a_r \mid s) \rangle \\ & + \lambda \left(\tfrac{1}{2}\|u_{r,1}(a_r \mid s) - u^{\text{ref}}_{r,1}(a_r \mid s)\|_2^2 - \tfrac{1}{2}\eta^2 \right) \end{aligned} \tag{19}$$

Let $u^*_{r,1}(a_r \mid s)$ and λ^* be the primal and dual optima. The KKT conditions are:

$$f_1(u^*_{r,1}(a_r \mid s)) \le 0 \qquad \text{(Primal feasibility)} \tag{20}$$

$$\lambda^* \ge 0 \qquad \text{(Dual feasibility)} \tag{21}$$

$$\lambda^* f_1(u^*_{r,1}(a_r \mid s)) = 0 \qquad \text{(Complementary slackness)} \tag{22}$$

$$-\nabla_a Q_\phi(s,\, a_1) + \lambda^*(u^*_{r,1}(a_r \mid s) - u^{\text{ref}}_{r,1}(a_r \mid s)) = 0 \qquad \text{(Stationarity)} \tag{23}$$

Assuming $\nabla_a Q_\phi(s,\, a_1) \neq 0$, stationarity equation 23 requires $\lambda^* > 0$. Complementary slackness equation 22 then forces the constraint to be active:

$$\|u^*_{r,1}(a_r \mid s) - u^{\text{ref}}_{r,1}(a_r \mid s)\|_2 = \eta \tag{24}$$

Taking the norm of the stationarity condition gives $\lambda^*\eta = \|\nabla_a Q_\phi(s,\, a_1)\|_2$, so $\lambda^* = \|\nabla_a Q_\phi(s,\, a_1)\|_2/\eta$. Substituting λ^* back into the stationarity condition yields:

$$u^*_{r,1}(a_r \mid s) = u^{\text{ref}}_{r,1}(a_r \mid s) + \eta \frac{\nabla_a Q_\phi(s, a_1)}{\|\nabla_a Q_\phi(s, a_1)\|_2} \tag{25}$$

□

A.2 Equivalence of Eulerian and Mean Flow Policies

In this section, we elucidate the equivalence between Mean Flow Policies [Zhan et al., 2026] and the Eulerian Policy in eq. (5). We begin by stating the Mean Flow Policy and its loss gradient with respect to parameters θ.

$$\mathcal{L}_{\text{MF}} = \mathbb{E}\left[\left|u^\theta_{r,t}(a_r \mid s) - \texttt{sg}\left(u^\theta_{r,t}(a_r \mid s) + (t-r)\frac{\mathrm{d}}{\mathrm{d}r}u^\theta_{r,t}(a_r \mid s)\right)\right|^2\right] \tag{26}$$

$$\nabla_\theta \mathcal{L}_{\text{MF}} = 2\mathbb{E}\left[\nabla_\theta u^\theta_{r,t}(a_r \mid s)^T\left(u^\theta_{r,t}(a_r \mid s) - \texttt{sg}\left(u^\theta_{r,t}(a_r \mid s) + (t-r)\frac{\mathrm{d}}{\mathrm{d}r}u^\theta_{r,t}(a_r \mid s)\right)\right)\right],$$

where the expectation is taken with respect to $(r, t, p_r(a_r \mid z, s))$. Now let us recall the Eulerian objective with a flow map policy parametrization $X_{r,t}(a_r \mid s) = a_r + (t-r)u^\theta_{r,t}(a_r \mid s)$ with explicit parameters θ for the average velocity:

$$\mathcal{L}_{\text{EPD}}(\theta) = \mathbb{E}\left[\left|\partial_r X_{r,t}(a_r \mid s) + \texttt{sg}\left(\nabla X_{r,t}(a_r \mid s)u^\theta_{r,r}(a_r \mid s)\right)\right|^2\right], \tag{27}$$

Let us examine the terms inside the squared norm and remove the stop-gradient operator sg. We compute the partial derivative with respect to the start time r:

$$\partial_r X_{r,t}(a_r \mid s) = -u^\theta_{r,t}(a_r \mid s) + (t-r)\partial_r u^\theta_{r,t}(a_r \mid s). \tag{28}$$

Plugging this back into the Eulerian objective, we have,

$$\mathcal{L} = \mathbb{E}\left[\left|\underbrace{-u^\theta_{r,t}(a_r \mid s)}_{T_1} + \underbrace{(t-r)\partial_r u^\theta_{r,t}(a_r \mid s) + \nabla X_{r,t}(a_r \mid s)u^\theta_{r,r}(a_r \mid s)}_{T_2}\right|^2\right]. \tag{29}$$

Applying a stop-gradient to T_2 and taking parameter gradients, Plugging this back into the Eulerian objective, we have,

$$\nabla_\theta \mathcal{L}(\theta) = 2\mathbb{E}\left[u^\theta_{r,t}(a_r \mid s) - \nabla_\theta u^\theta_{r,t}(a_r \mid s)\cdot\left(u^\theta_{r,t}(a_r \mid s) + \texttt{sg}\left(T_2\right)\right)\right]. \tag{30}$$

Now expanding the spatial gradient term in T_2, that is $\nabla X_{r,t}(a_r \mid s)u^\theta_{r,r}(a_r \mid s)$:

$$\nabla X_{r,t}(a_r \mid s)u^\theta_{r,r}(a_r \mid s) = u^\theta_{r,r}(a_r \mid s) + (t-r)\nabla u^\theta_{r,t}(a_r \mid s)u^\theta_{r,r}(a_r \mid s). \tag{31}$$

Now by invoking the tangent condition and replacing $u^\theta_{r,r}$ with the ground truth instantaneous velocity v^* we can expand T_2 have

$$T_2 = (t-r)\partial_r u^\theta_{r,t}(a_r \mid s) + v^*_r(a_r \mid s) + (t-r)\nabla u^\theta_{r,t}(a_r \mid s)v^*_r(a_r \mid s).$$

Rearranging terms and grouping $(t-r)$ terms, we notice the total derivative $\mathrm{d}/\mathrm{d}r$ corresponds exactly to T_2. We now leverage and rewrite eq. (30) succinctly:

$$\nabla_\theta \mathcal{L}(\theta) = 2\mathbb{E}\left[\nabla_\theta u^\theta_{r,t}(a_r \mid s)^T\left(u^\theta_{r,t}(a_r \mid s) - \texttt{sg}\left(u^\theta_{r,t}(a_r \mid s) + (t-r)\frac{\mathrm{d}}{\mathrm{d}r}u_{r,t}(a_r \mid s)\right)\right)\right].$$

This loss gradient matches the Mean Flow Policies' loss gradient, with the main distinction being the usage of the ground truth velocity v^*_r as opposed to the network's prediction $u^\theta_{r,r}$. Furthermore, the instantaneous velocity constraint is equivalent to the diagonal loss of eq. (3). This demonstrates that Mean Flow policies [Zhan et al., 2026] are not an independent paradigm, but rather a specific instantiation of the broader Eulerian Policy Distillation framework.

B Successful Rollouts

We visualize successful rollouts from the trained FMQ policy across all 12 evaluation environments. Each figure shows uniformly-spaced frames from a single episode that achieves the task goal.

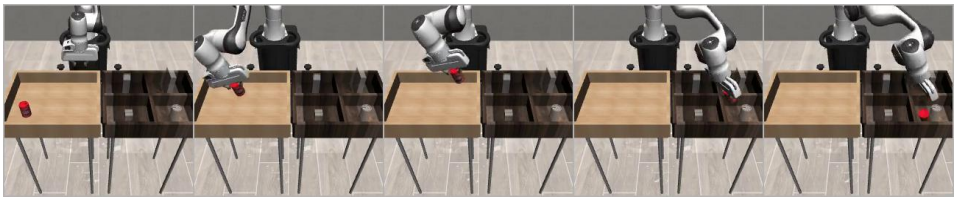

Figure 5: `can` (Robomimic). Pick up a can from the table and place it into the bin.

Figure 6: `square` (Robomimic). Pick up a square nut and fit it onto a peg.

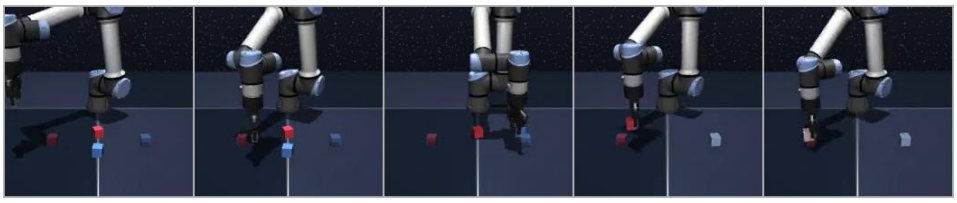

Figure 7: `cube-double-task3` (OGBench). Rearrange 2 cubes to target positions.

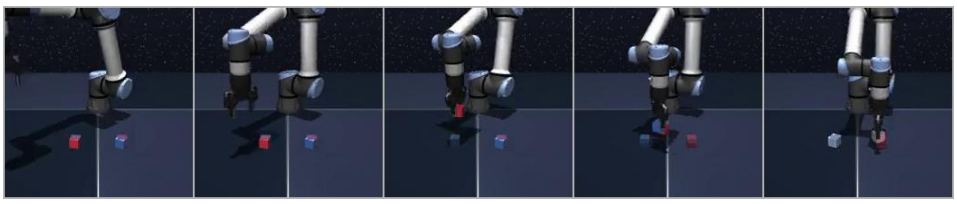

Figure 8: `cube-double-task4` (OGBench). Swap the positions of 2 cubes.

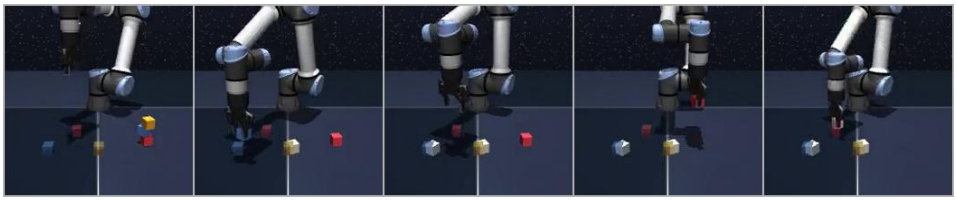

Figure 9: `cube-triple-task3` (OGBench). Unstack 3 cubes and place them at separate target positions.

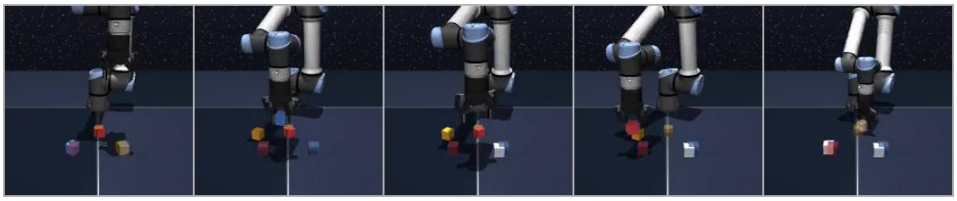

Figure 10: `cube-triple-task4` (OGBench). Cyclically permute 3 cubes to new positions.

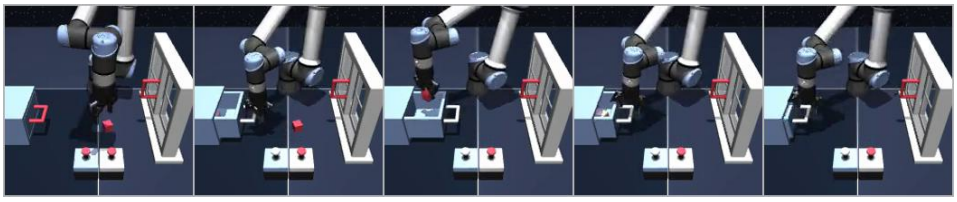

Figure 11: `scene-task4` (OGBench). Unlock the drawer button, open the drawer, and place the cube inside.

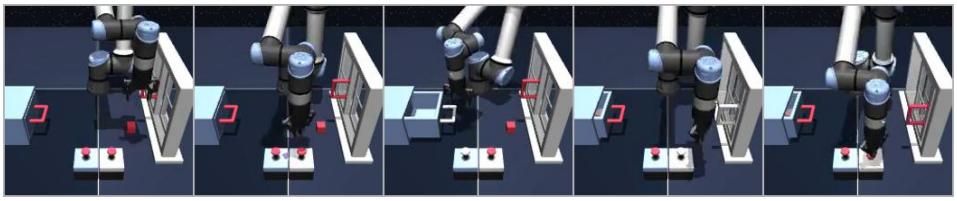

Figure 12: `scene-task5` (OGBench). Place the cube in the drawer and open the window.

Figure 13: `humanoidmaze-medium-task3` (OGBench). Navigate a humanoid to a goal in a medium maze.

Figure 14: `humanoidmaze-medium-task4` (OGBench). Navigate a humanoid to a distant goal in a medium maze.

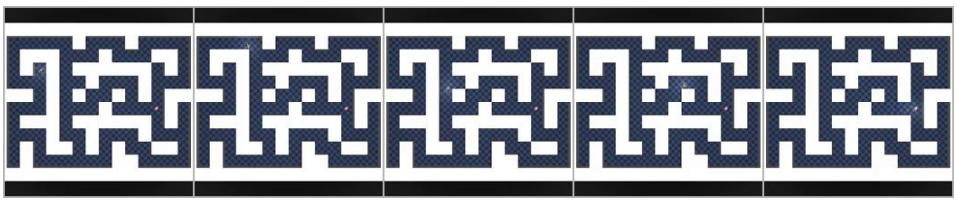

Figure 15: `antmaze-giant-task4` (OGBench). Navigate an ant to a goal across a giant maze.

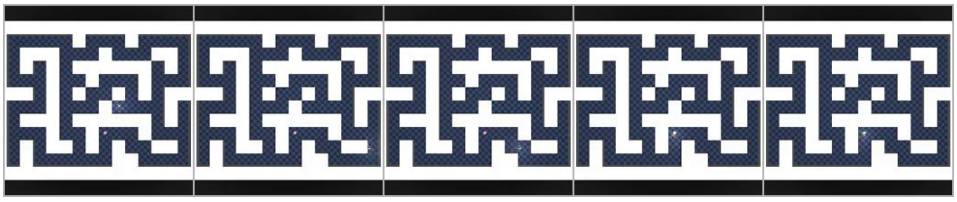

Figure 16: `antmaze-giant-task5` (OGBench). Navigate an ant to a nearby goal in a giant maze.

C Algorithms

Algorithm 1 FLOW MAP Q-GUIDANCE (FMQ)

Require: Offline policy $u^{\text{off}}_{r,1}$, online policy $u^{\theta}_{r,1}$, critics Q_{ϕ_1}, Q_{ϕ_2}, buffer $\mathcal{D}$

for each environment step **do**
  $a_1 \leftarrow a_0 + u^{\theta}_{0,1}(a_0|s)$, $a_0 \sim \mathcal{N}(0, I)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s, a_1, r, s')\}$
  Sample batch from $\mathcal{D}$; update critics via Eq. 8
  $r \sim \mathcal{U}[0, 1)$; $a_0 \sim \mathcal{N}(0, I)$; $a_r \leftarrow (1-r)a_0 + r\, a_{\text{data}}$
  $a_1 \leftarrow a_r + (1-r)\, u^{\text{off}}_{r,1}(a_r|s)$
  $g \leftarrow \nabla_a Q_{\phi_1}(s, a_1)/(\|\nabla_a Q_{\phi_1}(s, a_1)\|_2 + \kappa_1)$
  $\eta_{\text{eff}} \leftarrow \eta/(1 + \beta\, \tilde{\delta}_{\text{critic}})$ ▷ Eq. 13
  $\theta \leftarrow \theta - \alpha\, \nabla_\theta \|u^{\theta}_{r,1}(a_r|s) - \text{sg}(u^{\text{off}}_{r,1}(a_r|s) + \eta_{\text{eff}}\, g)\|^2$
end for

Algorithm 2 Q-GUIDED BEAM SEARCH (QGBS)

Require: Flow map $X^{\theta}_{r,1}$, critic Q_ϕ, state s, beam M, steps K, branches B, SNR ρ, step size η

$t' \leftarrow \rho/(1+\rho)$
Sample $\{a^m_0\}_{m=1}^{M} \sim \mathcal{N}(0, I)$; $a^m_1 \leftarrow a^m_0 + u^{\theta}_{0,1}(a^m_0|s)$ for all m
for $k = 1, \ldots, K$ **do**
  for $m = 1, \ldots, M$ and $b = 1, \ldots, B$ **do**
    $\varepsilon^{mb} \sim \mathcal{N}(0, I)$
    $\hat{a}^{mb}_1 \leftarrow X^{\theta}_{t',1}\big(t'\, a^m_1 + (1-t')\, \varepsilon^{mb} \mid s\big)$ ▷ Re-noise & complete
  end for
  $\{a^m_1\}_{m=1}^{M} \leftarrow \text{Top-}M\big(\{\hat{a}^{mb}_1\}_{m,b};\ Q_\phi(s, \hat{a}^{mb}_1)\big)$ ▷ Select best M of $M{\cdot}B$
  $a^m_1 \leftarrow a^m_1 + \eta\, \nabla_a Q_\phi(s, a^m_1)/\|\nabla_a Q_\phi(s, a^m_1)\|_2$ for all m ▷ Thm. 3.2
end for
return $a_1^{\arg\max_m Q_\phi(s, a^m_1)}$

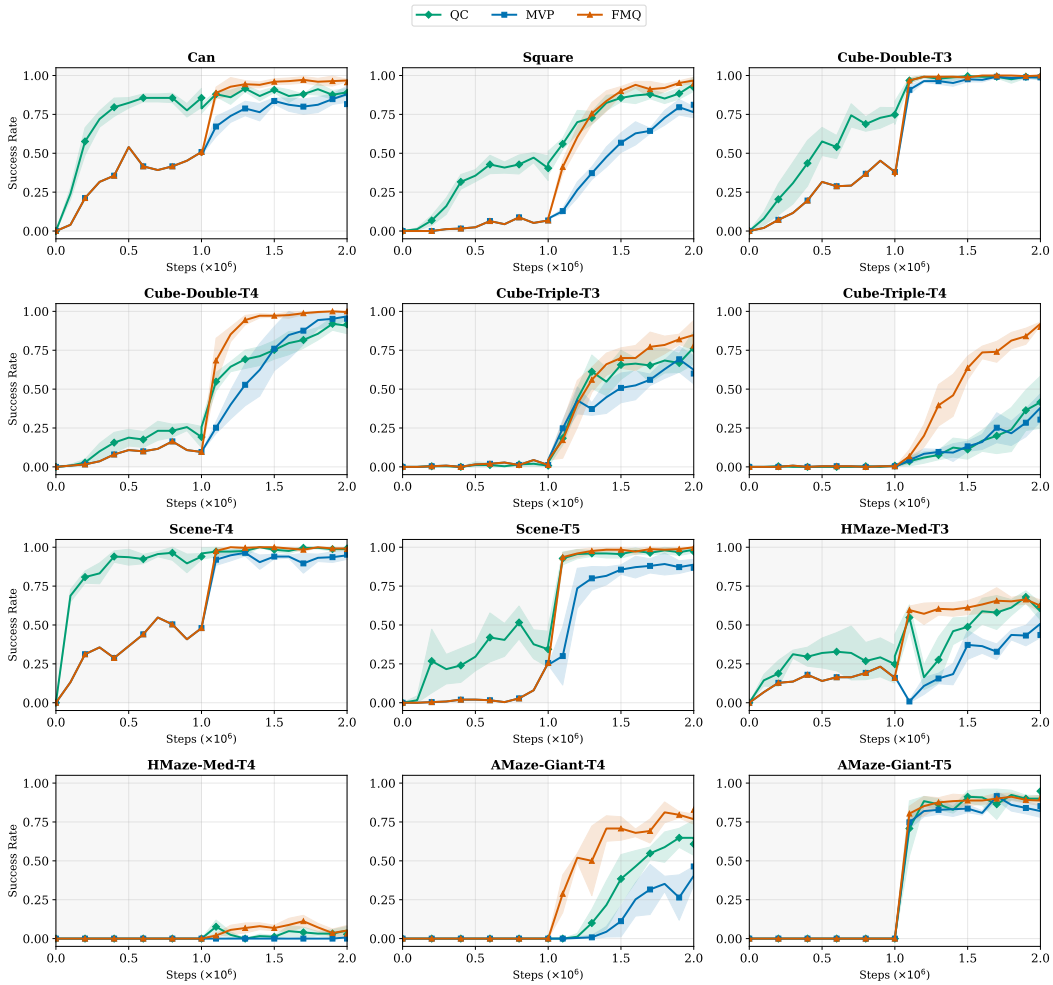

Figure 17: Offline-to-online learning curves for QC, MVP, and FMQ on all environments. All methods perform 1M offline followed by 1M online steps. Shaded regions indicate 95% CIs over 5 seeds.

D Training Curves

Figure 17 extends fig. 2 to all 12 environments. On the simpler manipulation tasks (`can`, `square`, `cube-dbl`), all methods converge to near-perfect success, but FMQ reaches this level earlier. The advantage becomes more pronounced on the harder tasks: on `cube-trl-t4`, FMQ reaches 0.88 while MVP plateaus at 0.32; on `amaze-gnt-t4`, FMQ achieves 0.80 versus 0.42 for MVP. For locomotion (`hmaze`, `amaze`), the Q-gradient signal is particularly beneficial under sparse rewards, where best-of-N selection provides a weaker learning signal.

E Trust-Region Convergence

Figure 18 extends the convergence analysis of fig. 4 to all 12 environments. We track the action displacement $\|u^{\text{on}}_{r,1} - u^{\text{off}}_{r,1}\|_2$ between the online and frozen offline flow map policies throughout online training. At the onset of fine-tuning (1M steps), both policies coincide and the displacement is near zero. As training progresses, the trust-region loss in eq. (12) drives $u^{\text{on}}_{r,1}$ toward $u^{\text{off}}_{r,1} + \eta_{\text{eff}}\,\hat{g}$, causing the displacement to grow monotonically until it stabilizes near η_{eff}. The orange curve (right axis) shows the implied Q-uncertainty $\tilde{\sigma}_Q = (\eta_{\text{eff}}^{-1} - 1)/\beta$, which decreases as the critic

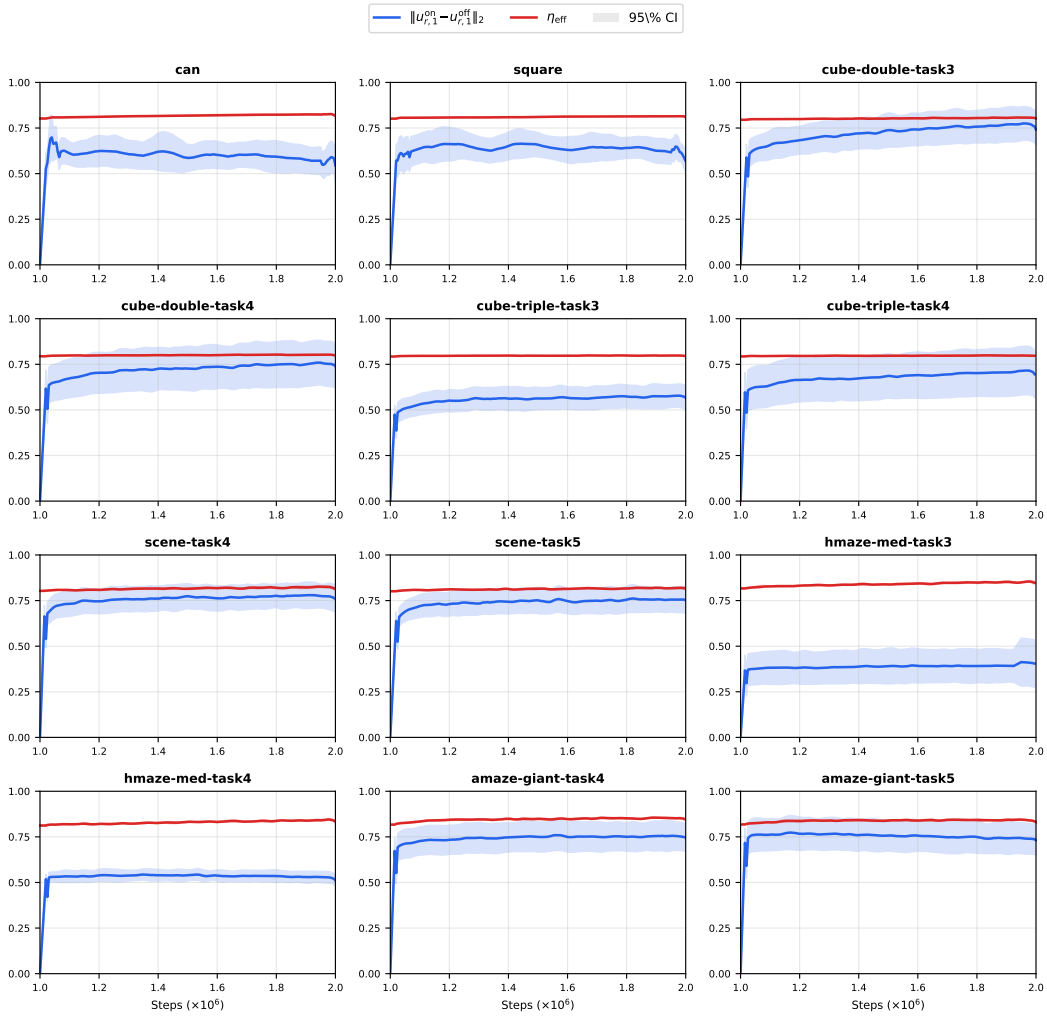

Figure 18: Trust-region convergence for FMQ (β=0.3) across all 12 environments. Blue: action displacement $\|u^{\text{on}}_{r,1} - u^{\text{off}}_{r,1}\|_2$. Red dashed: adaptive trust-region radius η_{eff}. Orange dotted (right axis): implied Q-uncertainty $\tilde{\sigma}_Q = (\eta_{\text{eff}}^{-1} - 1)/\beta$.

becomes more confident—automatically tightening the trust region and confirming that the adaptive mechanism prevents overshooting in low-confidence regions.

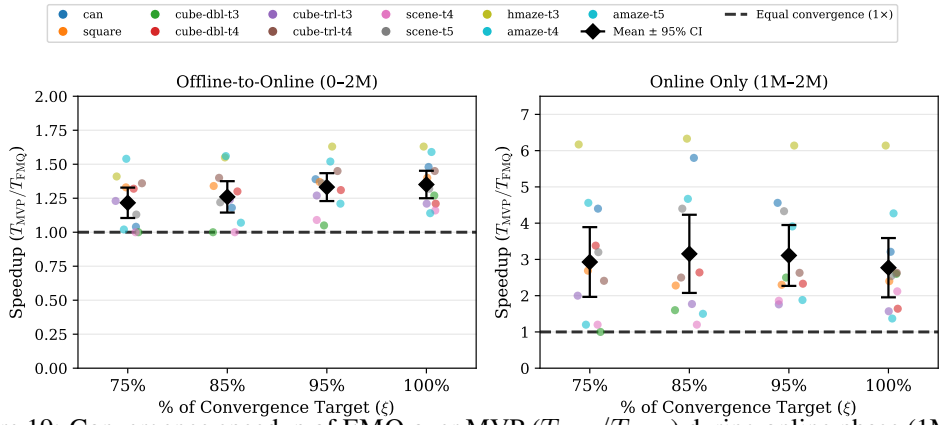

Figure 19: Convergence speedup of FMQ over MVP ($T_{\text{MVP}}/T_{\text{FMQ}}$) during online phase (1M–2M steps). Each dot represents one environment; black diamonds show the mean with 95% CI. The dashed line marks equal convergence speed ($1\times$).

Table 4: Speedup of FMQ over MVP ($T_{\text{MVP}}/T_{\text{FMQ}}$) measuring time to reach a fraction of the convergence target ξ (per seed, averaged over 5 seeds). **Left**: full training (0–2M). **Right**: online phase only (1M–2M). Values > 1 indicate FMQ is faster.

	Full (0–2M)				**Online (1M–2M)**			
Environment	75%	85%	95%	100%	75%	85%	95%	100%
`can`	1.04	1.18	1.39	1.48	4.40	5.80	4.56	3.21
`square`	1.33	1.34	1.37	1.40	2.69	2.28	2.30	2.40
`cube-double-task3`	1.00	1.00	1.05	1.27	1.00	1.60	2.50	2.60
`cube-double-task4`	1.32	1.30	1.31	1.21	3.38	2.64	2.33	1.64
`cube-triple-task3`	1.23	1.24	1.27	1.21	2.00	1.77	1.76	1.57
`cube-triple-task4`	1.36	1.40	1.45	1.45	2.41	2.50	2.63	2.63
`scene-task4`	1.00	1.00	1.09	1.16	1.20	1.20	1.86	2.12
`scene-task5`	1.13	1.22	1.36	1.32	3.20	4.40	4.33	2.54
`hmaze-med-task3`	1.41	1.55	1.63	1.63	6.17	6.33	6.14	6.14
`amaze-giant-task4`	1.54	1.56	1.52	1.59	4.56	4.67	3.91	4.27
`amaze-giant-task5`	1.02	1.07	1.21	1.14	1.20	1.50	1.88	1.37
Average	**1.22**	**1.26**	**1.33**	**1.35**	**2.93**	**3.15**	**3.11**	**2.77**
95% CI	[1.10, 1.33]	[1.14, 1.37]	[1.23, 1.43]	[1.25, 1.45]	[1.97, 3.89]	[2.08, 4.23]	[2.27, 3.95]	[1.95, 3.59]

F Speedup Analysis

Figure 19 visualizes the per-environment convergence speedup of FMQ over MVP during the online phase (1M–2M), complementing the discussion in section 4.2. For each threshold $\xi \in 75\%, 85\%, 95\%, 100\%$ of the shared convergence target, we plot the ratio $T_{\text{MVP}}/T_{\text{FMQ}}$. FMQ is faster than MVP on every environment at every threshold (all points above $1\times$), with average speedups of 2.8–3.2$\times$. The full per-environment breakdown including both full-training and online-only phases is provided in table 4.

Table 5: QGBS on FMQ. SNR= 1.5, $\eta = 0.3$. Columns grouped by K; sub-columns $\{B, M\}$. Success rate (mean $\pm$ std, 5 seeds, 50 eps). Best per row in **bold**.

Environment	$K=0$	$K=1$				$K=2$	
	$\{1,32\}$	$\{1,16\}$	$\{2,8\}$	$\{4,4\}$	$\{4,16\}$	$\{1,16\}$	$\{4,4\}$
can	0.96 ± 0.04	0.97 ± 0.02	0.96 ± 0.04	0.97 ± 0.03	0.94 ± 0.04	0.95 ± 0.03	$\mathbf{0.98 \pm 0.03}$
square	0.94 ± 0.02	0.94 ± 0.03	$\mathbf{0.96 \pm 0.04}$	0.95 ± 0.04	$\mathbf{0.96 \pm 0.02}$	0.94 ± 0.04	0.94 ± 0.03
cdp3	$\mathbf{1.00 \pm 0.00}$	0.99 ± 0.01	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$
cdp4	0.98 ± 0.02	0.99 ± 0.03	0.99 ± 0.01	$\mathbf{1.00 \pm 0.00}$	0.98 ± 0.02	0.99 ± 0.03	0.99 ± 0.01
ctrp3	0.78 ± 0.10	0.82 ± 0.10	0.78 ± 0.06	$\mathbf{0.84 \pm 0.04}$	0.83 ± 0.07	$\mathbf{0.84 \pm 0.08}$	0.82 ± 0.08
ctrp4	$\mathbf{0.88 \pm 0.07}$	0.84 ± 0.06	$\mathbf{0.88 \pm 0.06}$	0.87 ± 0.05	0.82 ± 0.09	0.82 ± 0.04	0.84 ± 0.06
sc4	$\mathbf{1.00 \pm 0.00}$	$\mathbf{1.00 \pm 0.00}$	0.99 ± 0.01	0.99 ± 0.01	$\mathbf{1.00 \pm 0.00}$	0.99 ± 0.01	$\mathbf{1.00 \pm 0.00}$
sc5	0.98 ± 0.02	$\mathbf{1.00 \pm 0.00}$	0.99 ± 0.01	$\mathbf{1.00 \pm 0.00}$	0.99 ± 0.01	$\mathbf{1.00 \pm 0.00}$	0.98 ± 0.02
hm3	0.69 ± 0.04	0.70 ± 0.07	0.63 ± 0.04	0.58 ± 0.07	$\mathbf{0.72 \pm 0.11}$	0.68 ± 0.10	0.70 ± 0.07
hm4	0.06 ± 0.03	0.07 ± 0.04	0.10 ± 0.04	0.06 ± 0.03	$\mathbf{0.11 \pm 0.04}$	0.10 ± 0.05	0.10 ± 0.03
ag4	0.80 ± 0.06	0.78 ± 0.06	$\mathbf{0.86 \pm 0.10}$	0.77 ± 0.03	0.82 ± 0.04	0.75 ± 0.05	0.79 ± 0.03
ag5	$\mathbf{0.92 \pm 0.04}$	$\mathbf{0.92 \pm 0.04}$	0.90 ± 0.05	$\mathbf{0.92 \pm 0.05}$	0.90 ± 0.09	0.89 ± 0.03	0.90 ± 0.02
IQM	0.91 [0.89, 0.93]	0.92 [0.90, 0.93]	**0.93** [**0.91**, **0.95**]	**0.93** [**0.91**, **0.94**]	0.92 [0.90, 0.93]	0.90 [0.89, 0.92]	0.91 [0.90, 0.93]

G Inference-Time Beam Search

Table 5 provides the full per-environment breakdown of QGBS applied to the trained FMQ checkpoint, extending the aggregate IQM results reported in table 2. NFE $= M(1 + KB)$, where M is the number of initial candidates, K the number of renoising steps, and B the number of completions per candidate; K=0 reduces to standard best-of-M. The per-environment results confirm that the gains from renoising (K=1) are consistent across task domains—manipulation, multi-object rearrangement, and locomotion—with the most notable improvements on the harder maze tasks (hm3: $0.59 \to 0.72$, hm4: $0.07 \to 0.11$).

Table 6: Hyperparameters shared across all methods.

Parameter	**Value**
Optimizer	Adam
Learning rate	3×10^{-4}
Batch size	256
Discount (γ)	0.99
Target update (τ)	5×10^{-3}
UTD ratio	1
Offline / online steps	1M / 1M
Replay buffer	2M
Policy network	MLP, 4×512, GELU
Critic network	MLP, 4×512, GELU, LayerNorm
Critic ensemble	2 (double Q, mean agg.)
Fourier embedding	64 dim per time axis
Chunking horizon (H)	5
Eval interval / episodes	100K / 50

H Implementation Details

All methods share the same network architecture, critic algorithm, and training pipeline to ensure a controlled comparison. The policy is parameterized as a time-conditioned velocity field u_θ: a 4-layer MLP with 512 hidden units and GELU activations. Scalar flow times are lifted to 64-dimensional sinusoidal Fourier embeddings before concatenation with the observation and noisy action. The critic follows the clipped double Q-learning framework [Fujimoto et al., 2018]: an ensemble of two Q-networks with the same MLP architecture (with LayerNorm) trained against a shared Bellman target using Polyak-averaged target networks (τ=0.005). All methods use action chunking (H=5), a replay buffer of 2M transitions, and are trained for 1M offline followed by 1M online steps with UTD ratio 1, Adam (lr=3×10^{-4}), and batch size 256. Full shared hyperparameters are in table 6.

QC [Li et al., 2025] trains a standard CFM velocity field $v_\theta(a_t, t \mid s)$ with the straight-line interpolation objective. At inference, the ODE is integrated from t=0 to t=1 with 10 Euler steps, producing

Table 7: Inference procedures. NFE = network forward evaluations per action.

Method	**Action selection**	**Steps**	N	**NFE**
QC	Best-of-N (Euler)	10	32	320
MVP	Best-of-N (flow map)	1	32	32
MVP + QGBS (ours)	QGBS	K	M	$K \cdot M$
FMQ (ours)	Best-of-N (flow map)	1	32	32
FMQ + QGBS (ours)	QGBS	K	M	$K \cdot M$

32 candidates scored by the critic (best-of-N, 320 NFE total).MVP [Zhan et al., 2026] replaces multi-step Euler integration with a single-step flow map (K=1) that directly predicts the average velocity $u_{r,t}(a_r \mid s)$ over $[r, t]$. The network takes as input $[\mathbf{s}, \mathbf{x}_r, \text{Fourier}(r), \text{Fourier}(t), \text{Fourier}(t_c), \mathbf{a}_c]$ where $(t_c, \mathbf{a}_c)$ form a conditioning axis for stochastic action generation. Training uses a progressive curriculum: diagonal-only CFM (r=t) for 5K steps, then the interval $[r, t]$ is annealed to the full range over 50K steps, and the conditioning axis is introduced after 10K steps with $P(t_c{=}0){=}0.5$. At inference, a single forward pass generates each candidate and best-of-32 selection is applied (32 NFE). FMQ shares the same offline pretraining as MVP. During the online phase, it switches to the trust-region Q-gradient objective described in section 3.3: the offline flow map is frozen as the reference $u^{\text{off}}_{r,1}$, the Q-gradient is ℓ_2-normalized, and the trust-region radius η_{eff} adapts per sample via Q-ensemble disagreement (β=0.3, cf. eq. (13)). At inference, FMQ uses the same best-of-32 flow map selection as MVP (32 NFE). QGBS applies the Q-guided beam search of section 3.4 at inference time without additional training. Starting from a trained flow map, actions are diversified via SNR-based renoising and refined over K beam steps using M candidates, with the trust-region projection applied at each iteration. Steering strength is controlled by λ and actions are clipped via a straight-through estimator. Cost: $M(1 + KB)$ NFE per action. Inference procedures and computational costs are summarized in table 7. All training experiments were run on NVIDIA A100-SXM4-80GB GPUs. A full training run takes approximately 4 hours per seed. Inference-time steering evaluations were conducted on NVIDIA RTX 6000 Ada Generation GPUs (48 GB VRAM).