FLOW-LAG: SENSITIVITY-AWARE PRIMAL-DUAL OPTIMIZATION FOR SAFE FLOW POLICIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Flow-based policies can represent expressive action distributions and have recently shown strong performance in reinforcement learning. However, enforcing safety constraints for flow-based policies remains challenging. Most flow-policy optimization methods optimize surrogate objectives defined over the internal flow process, such as flow-matching or path-space objectives, rather than the deployed endpoint policy directly. This creates a surrogate-endpoint mismatch in which the policy is optimized using surrogate objectives, while safety is ultimately determined by the endpoint actions executed in the environment. We show that this mismatch can substantially weaken safety correction in primal-dual reinforcement learning. Specifically, increasing the Lagrange multiplier can induce substantial changes in the flow policy while yielding only limited reductions in cost. To address this issue, we propose **Flow-Lag**, a sensitivity-aware primal-dual framework that propagates endpoint cost information through the flow dynamics to measure the safety response of the actual policy update. Flow-Lag uses this response to adapt the multiplier and calibrate the policy update, directly linking primal-dual correction to its endpoint effect. Across Safety-Gymnasium navigation and velocity-constrained Safe MuJoCo benchmarks, Flow-Lag achieves improved reward-safety performance over representative flow-based and conventional safe RL baselines.

1 INTRODUCTION

Flow-based policies have recently emerged as an expressive policy class in reinforcement learning (RL), enabling flexible modeling of complex action distributions (Zhang et al., 2026; Chen et al., 2025; McAllister et al., 2026; Ho et al., 2019). Unlike conventional parametric policies that directly specify an action distribution, flow policies generate actions by transporting samples from a simple base distribution through a learned sequence of intermediate dynamics. This expressive transport-based parameterization has made flow policies increasingly attractive for challenging continuous-control problems. Despite rapid progress in flow-based RL, reliable safety enforcement for such policies remains largely underexplored.

Among established approaches to constrained RL, primal–dual optimization is one of the most widely adopted, using a Lagrange multiplier to balance task reward against constraint violations during policy optimization (Achiam et al., 2017; Stooke et al., 2020; Wu et al., 2024). However, extending this paradigm to flow policies is nontrivial. For standard stochastic policies, primal-dual optimization acts directly on the policy likelihood, so increasing the cost penalty reshapes the distribution over executed actions toward lower-cost behavior. For flow policies, however, this direct correspondence is weakened because the likelihood of the deployed endpoint action is generally intractable, requiring policy optimization to rely on flow-matching, path-space, or related surrogate objectives defined over the internal transport process (McAllister et al., 2026; Chen et al., 2025; Zhang et al., 2026). Therefore, the constraint penalty imposed by the Lagrange multiplier acts first on the surrogate optimization process and only indirectly affects the deployed endpoint behavior, as illustrated in Fig. 1. Consequently, increasing the cost penalty can substantially alter the internal flow update without producing a commensurate reduction in endpoint cost. When the constraint remains violated, the primal-dual update continues to increase multiplier λ, causing the cost term to increasingly dominate the policy objective while providing little additional safety improvement. As

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.

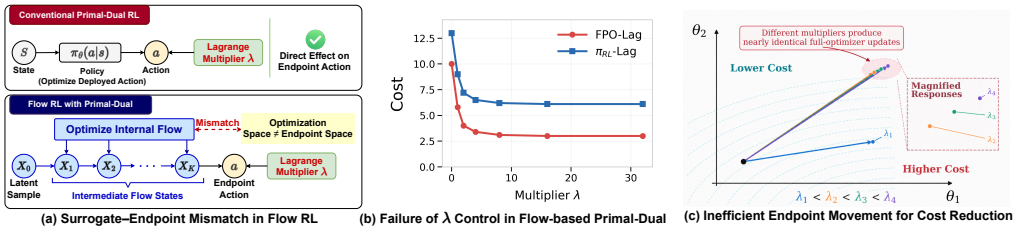

Figure 1: Surrogate-endpoint mismatch in primal–dual flow RL. (a) Optimization acts on the internal flow process, whereas safety is evaluated at the endpoint. (b) Increasing the Lagrange multiplier can yield diminishing endpoint cost reduction. (c) The induced endpoint update may align poorly with the local cost-descent direction.

a result, the optimization can become overly conservative and eventually unstable, while still failing to effectively reduce constraint violations.

To address this limitation, we propose **Flow-Lag**, a sensitivity-aware primal-dual framework for safe flow reinforcement learning. Flow-Lag propagates endpoint cost information backward through the flow dynamics to quantify how the actual policy optimizer update affects deployed safety behavior. This endpoint response provides the missing feedback between surrogate optimization and endpoint constraint satisfaction. Flow-Lag then uses this feedback in two complementary ways: it adapts the Lagrange multiplier according to the observed effectiveness of previous safety corrections, and it calibrates the magnitude of a cost-decreasing policy update before deciding whether a further increase in the multiplier is warranted. In this way, Flow-Lag closes the feedback loop between constraint violation, flow-policy optimization, and endpoint behavior, enabling more effective safety correction for flow policies.

We summarize our contributions as follows: (1) We identify a fundamental optimization–behavior mismatch in primal-dual safe flow RL, where stronger constraint penalty does not necessarily translate into effective endpoint cost reduction. (2) We propose Flow-Lag, a sensitivity-guided primal-dual framework that uses endpoint behavioral feedback to improve safety optimization for flow policies. (3) We conduct extensive experiments on Safe MuJoCo velocity-control and Safety-Gymnasium navigation benchmarks, where Flow-Lag achieves improved reward–safety performance and training stability over flow-based safety baselines and representative safe RL methods.

2 Related Work

2.1 Flow-based Policy in Reinforcement Learning

Flow matching learns a velocity field that transports samples from a simple base distribution toward a target distribution (Liu et al., 2026; Lipman et al., 2024; Dao et al., 2023; Gat et al., 2024). Early flow RL work primarily focused on enabling policy-gradient optimization despite the intractable endpoint likelihood of flow policies (Liu et al., 2022; Tong et al., 2023; Albergo & Vanden-Eijnden, 2022; McAllister et al., 2026). ReinFlow injects learnable stochasticity into the deterministic flow process, converting denoising into a tractable Markov process for likelihood-based policy optimization (Zhang et al., 2026). Similarly, π_{RL} introduces stochastic flow formulations for online reinforcement learning of flow-based vision-language-action policies (Chen et al., 2025). Rather than recovering an explicit likelihood, Flow Policy Optimization (FPO) constructs an advantage-weighted policy surrogate directly from conditional flow matching losses, thereby enabling PPO-style optimization without evaluating the endpoint action density (McAllister et al., 2026). Safety for flow-based policies remains considerably less explored. SafeFlowMatcher combines flow matching with control barrier functions to enforce safety during trajectory generation (Yang et al., 2026). Safe Flow Q-Learning extends flow-based offline RL with a reachability-inspired safety value function and a one-step flow actor, targeting offline constrained control (Tayal et al., 2026). In this work, we investigate safe reinforcement learning with primal-dual optimization for flow-based policies, with a particular focus on understanding and addressing the mismatch between internal flow optimization and the safety behavior of the deployed policy.

2.2 PRIMAL-DUAL METHODOLOGY

Safe reinforcement learning has been widely studied in safety-related control problems (Xu et al., 2021; Chow et al., 2018; Yang et al., 2022; Ray et al., 2019; Dalal et al., 2018). Primal-dual methods provide a general framework for constrained reinforcement learning by formulating safety requirements as constraints in a constrained Markov decision process (CMDP) (Achiam et al., 2017; Tessler et al., 2018; Stooke et al., 2020; Ding et al., 2020). Subsequent studies have improved the stability and constraint satisfaction of primal-dual learning through PID-style multiplier control, conservative natural-gradient updates, adaptive primal-dual learning rates, and related regularization or penalty mechanisms (Stooke et al., 2020). Despite these advances, primal-dual optimization remains sensitive to the coupling between the dual update and the induced primal policy response, and inappropriate multiplier dynamics can lead to constraint oscillation and excessive conservatism (Chen et al., 2024). More recent work has explored sensitivity-aware constrained optimization, such as CSPO, which uses local constraint sensitivity to improve policy correction (Belouadah et al., 2026), while ALGD extends primal-dual optimization to diffusion policies through an augmented Lagrangian formulation (Cheng et al., 2026). In contrast, our work focuses on the surrogate-endpoint mismatch specific to flow policies and introduces endpoint sensitivity feedback to improve how primal-dual constraint penalty translates into deployed action.

3 PRELIMINARIES

3.1 CMDPS AND PRIMAL-DUAL OPTIMIZATION

We consider a constrained Markov decision process (CMDP) problem (Altman, 2021) defined by the tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, c, \rho_0, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the state and action spaces, P is the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ is the cost function, ρ_0 is the initial-state distribution, and $\gamma \in (0, 1)$ is the discount factor. Given a policy π, let $\tau = (s_0, a_0, s_1, a_1, \ldots)$ denote a trajectory generated by $s_0 \sim \rho_0$, $a_t \sim \pi(\cdot \mid s_t)$, and $s_{t+1} \sim P(\cdot \mid s_t, a_t)$. Its expected discounted reward and cost are

$$J_R(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right], \qquad J_C(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t)\right]. \tag{1}$$

The safe reinforcement learning objective is

$$\max_{\pi} \; J_R(\pi) \qquad \text{s.t.} \qquad J_C(\pi) \leq d, \tag{2}$$

where d denotes the prescribed safety budget. We further denote the reward and cost action-value functions under π by $Q_R^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \,|\, s_0 = s,\ a_0 = a\right], Q_C^\pi(s, a) = \mathbb{E}_\pi\left[\sum_{t=0}^{\infty} \gamma^t c(s_t, a_t) \,|\, s_0 = s,\ a_0 = a\right]$.

In practical policy optimization, we consider a differentiable π_θ parameterized by θ. Primal-dual methods address this constrained problem through the Lagrangian (Paternain et al., 2019)

$$\mathcal{L}(\theta, \lambda) = J_R(\pi_\theta) - \lambda\big(J_C(\pi_\theta) - d\big), \qquad \lambda \geq 0, \tag{3}$$

where λ controls the penalty on constraint violation. Primal-dual optimization alternates between updating the policy parameters under the current multiplier and updating the multiplier through projected dual ascent (Mangasarian, 1994)

$$\lambda_{k+1} = \big[\lambda_k + \eta_\lambda\big(J_C(\pi_{\theta_k}) - d\big)\big]_+, \tag{4}$$

where k indexes training iterations, $\eta_\lambda > 0$ is the dual learning rate, and $[x]_+ = \max\{x, 0\}$. Thus, constraint violations increase the safety penalty, while costs below the budget allow it to decrease.

3.2 FLOW POLICIES

In this work, we instantiate π_θ as a flow-based policy using a continuous-time transport process that maps samples from a base distribution p_0 to actions (Chen et al., 2018; Lipman et al., 2022; Grathwohl et al., 2018). For each environment state s_t, we sample a base-noise variable $z_t \sim p_0$,

where p_0 denotes a simple base distribution such as $\mathcal{N}(0, I)$, and evolve a flow state x_σ according to the state-conditioned velocity field v_θ

$$\frac{dx_\sigma}{d\sigma} = v_\theta(x_\sigma, \sigma; s_t), \qquad x_0 = z_t, \qquad \sigma \in [0, 1], \tag{5}$$

where σ denotes the flow time. The deployed action is given by the endpoint of this transport process

$$z_t \sim p_0, \qquad a_t = x_1 = F_\theta^t(s_t, z_t), \qquad a_t \sim \pi_\theta(\cdot \mid s_t). \tag{6}$$

Here, F_θ denotes the transport map induced by integrating Eq. 5. Thus, the endpoint policy $\pi_\theta(\cdot \mid s_t)$ is the distribution induced by transporting $z_t \sim p_0$ through $F_\theta^t(s_t, \cdot)$. In practice, the continuous flow is evaluated through a finite-step numerical integration. For clarity, consider a K-step explicit Euler discretization

$$x_{j+1} = x_j + h\, v_\theta^j(x_j, \sigma_j; s_t), \qquad j = 0, \ldots, K-1, \tag{7}$$

where h denotes the integration step size, and the deployed action is $a_t = x_K$.

To characterize this propagation, consider a differentiable scalar endpoint quantity $\Phi(a_t)$. A perturbation to an intermediate flow velocity changes the next flow state and subsequently propagates through the remaining integration steps to the endpoint. This dependence can be computed efficiently using a discrete adjoint (Chen et al., 2018). Define

$$q_j \triangleq \nabla_{x_j} \Phi(a_t), \qquad a_t = x_K, \tag{8}$$

with terminal condition $q_K = \nabla_{a_t} \Phi(a_t)$. By the chain rule, the adjoint propagates backward according to

$$q_j = \left(I + h \frac{\partial v_\theta^j(x_j, \sigma_j; s_t)}{\partial x_j} \right)^\top q_{j+1}. \tag{9}$$

4 Methodology

4.1 Surrogate-Endpoint Mismatch in Flow Policy Optimization

Primal-dual methods regulate safety by increasing the Lagrange multiplier when the current policy violates the constraints, thereby placing greater weight on cost reduction during policy optimization. Their effectiveness therefore depends on whether the increased weighting of the cost term is translated into deployed action that actually reduces cost. For flow policies, however, the policy optimizer does not update the deployed endpoint action directly. Instead, it updates the internal velocity field through a surrogate objective defined along the flow trajectory, and the endpoint action changes only as a consequence of integrating these modified internal dynamics. As a result, a larger multiplier may induce a substantial change in the internal flow update, but this may not translate into a comparably large change in the endpoint action or its resulting cost.

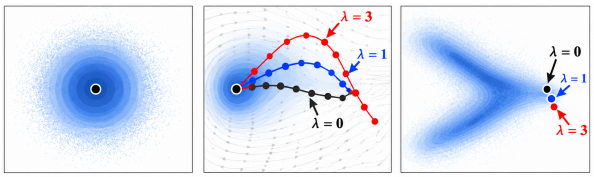

Figure 2: Illustration of surrogate–endpoint mismatch in the analytic flow example.

A simple dimensional observation helps explain this weakened correspondence. Consider a K-step flow policy with action dimension d_a. Around the current flow trajectory, the effect of small perturbations to the intermediate velocities on the endpoint action can be locally approximated by a linear mapping. This mapping takes a Kd_a-dimensional internal perturbation space to a d_a-dimensional endpoint-action space. Since its rank is at most d_a, the rank-nullity theorem implies that it has a nontrivial null space whenever $K > 1$. Hence, there exist directions in the internal velocity space that change the intermediate flow while producing no first-order change in the endpoint action. More generally, an internal flow update can contain components that have only a weak effect on the deployed endpoint action. Figure 2 illustrates this effect in an analytically tractable example. As λ increases, the intermediate flow trajectories change substantially, while the endpoint actions remain close. This behavior is consistent with multiplier-induced updates containing directions that strongly affect the internal flow but only weakly affect the endpoint action.

4.2 Endpoint Sensitivity of the Actual Policy Update

Although the magnitude of a surrogate update is not a reliable indicator of its effect on safety, we can still quantify which internal flow changes influence the deployed action, i.e., the sensitivity of endpoint cost to the intermediate flow velocities.

As shown in Eq. 9, the adjoint q_j characterizes the sensitivity of an endpoint quantity to the intermediate flow state x_j. However, the flow policy is parameterized through the velocity field v_θ, which determines the evolution of these intermediate states. We therefore further propagate the state sensitivity to the local flow velocity. Since $\partial x_{j+1}/\partial v_\theta^j = hI$, and the endpoint action $a_t = x_K$ is obtained by recursively integrating the intermediate velocities, $\Phi(a_t)$ implicitly depends on velocity v_θ^j. Therefore, applying the chain rule through x_{j+1} gives

$$\frac{d\Phi}{dv_\theta^j} = \left(\frac{\partial x_{j+1}}{\partial v_\theta^j}\right)^\top q_{j+1} = hq_{j+1}. \tag{10}$$

For safety optimization, the endpoint quantity of interest is the predicted cost of the deployed action. We therefore set $\Phi(a_t) = Q_C(s_t, a_t)$ with $a_t = F_\theta^t$, where the state s_t and cost critic Q_C are held fixed during the sensitivity evaluation. Accordingly, the generic adjoint becomes the cost adjoint

$$q_j^C \triangleq \nabla_{x_j} Q_C(s_t, F_\theta^t), \tag{11}$$

which follows the same backward recursion as Eq. 9. The corresponding flow velocity to endpoint cost sensitivity is

$$g_j^C \triangleq \nabla_{v_\theta^j} Q_C(s_t, F_\theta^t) = hq_{j+1}^C. \tag{12}$$

The sensitivity g_j^C characterizes how an arbitrary perturbation of the velocity at flow step j affects the endpoint cost value. However, to evaluate an actual policy update, we additionally need to determine how that update changes the intermediate velocities.

We next use this sensitivity to evaluate the update produced by the actual flow-policy optimizer. Starting from the current parameters θ_k, one complete policy-optimization update of the underlying flow-RL backbone produces a candidate parameter vector θ_k^+. We define the corresponding optimizer displacement as

$$\Delta\theta_k = \theta_k^+ - \theta_k, \qquad \theta_k(\alpha) = \theta_k + \alpha\Delta\theta_k. \tag{13}$$

Here, θ_k^+ is the full candidate proposed by the underlying flow-policy optimizer before response-aware calibration. The scalar α parameterizes movement along this realized optimizer direction $\alpha = 0$ recovers the current policy θ_k, $\alpha = 1$ gives the full optimizer candidate θ_k^+.

Along this update direction, the induced first-order change of the velocity at flow step j is

$$\delta v_j = \left.\frac{\partial v_\theta^j}{\partial \theta}\right|_{\theta=\theta_k^+} \Delta\theta_k. \tag{14}$$

To summarize the endpoint safety effect across sampled states and flow samples, we define the endpoint cost proxy $\bar{\mathcal{Q}}_C(\theta) = \mathbb{E}_{(s_t,z_t)}\left[Q_C(s_t, F_\theta^t)\right]$. The endpoint response of the actual optimizer update is defined as

$$S_k \triangleq \frac{d\,\bar{\mathcal{Q}}_C(\theta_k(\alpha))}{d\alpha} = \mathbb{E}_{(s_t,z_t)}\left[\sum_{j=0}^{K-1} (g_j^C)^\top \delta v_j\right]. \tag{15}$$

Eq. 15 measures how strongly the actual flow-policy update aligns with directions that affect endpoint safety. A negative response, $S_k < 0$, indicates that the candidate is locally cost-decreasing along the optimizer direction, whereas $S_k > 0$ indicates a locally cost-increasing direction. A small $|S_k|$ indicates that the optimizer produces little endpoint safety response, even when the internal flow update is large.

Lemma 1 (Endpoint Cost Decrease under Flow Sensitivity) *Let θ_k^+ denote the candidate produced by one complete flow-policy optimizer update and define $\Delta\theta_k = \theta_k^+ - \theta_k$. Suppose that the fixed-critic endpoint cost proxy $\bar{\mathcal{Q}}_C(\theta)$ has an L_C-Lipschitz continuous gradient along the segment between θ_k and θ_k^+. Then*

$$\bar{\mathcal{Q}}_C(\theta_k^+) - \bar{\mathcal{Q}}_C(\theta_k) \le S_k + \frac{L_C}{2}\|\Delta\theta_k\|^2. \tag{16}$$

4.3 Practical Response-Aware Flow Policy Optimization

The endpoint response S_k derived in Sec. 4.2 measures whether the actual policy update moves the deployed policy toward lower endpoint cost. The key question is then how this response should be used to improve primal–dual optimization. In particular, an insufficient safety correction may arise for two different reasons: the multiplier may still be able to strengthen the endpoint response, or the multiplier–response channel may have become locally ineffective as Sec. 4.1 analyzed.

Motivated by this distinction, we present **Flow-Lag**, a sensitivity-aware primal-dual framework for safe flow policy optimization. When the endpoint response remains sensitive to the multiplier, Flow-Lag adjusts the multiplier to strengthen the safety correction. When this relation becomes weak, further multiplier escalation provides little additional endpoint effect, so Flow-Lag suppresses the dual correction. If the current update is already cost-decreasing in this regime, Flow-Lag instead calibrates its step magnitude to exploit the effective direction.

Sensitivity-aware multiplier adaptation. Standard primal–dual methods adjust the multiplier according to constraint violation, which indicates whether additional constraint penalty is needed. However, for flow policies, violation alone does not reveal whether increasing this constraint penalty actually produces a stronger endpoint correction. Flow-Lag therefore augments the standard violation-based multiplier with an endpoint-response correction, yielding a revised multiplier $\widetilde{\lambda}_k$ that accounts for how strongly changes in constraint penalty translate into endpoint cost reduction.

Specifically, we first update the standard primal-dual multiplier λ using the observed mean episodic cost,

$$\lambda_{k+1} = \left[\lambda_k + \eta_\lambda \frac{\bar{J}_C - d}{d}\right]_+, \tag{17}$$

where $\bar{J}_C$ denotes the empirical mean episodic cost.

The standard multiplier λ_{k+1} reflects the constraint penalty suggested by the observed constraint violation, but does not account for how effectively this constraint penalty changes the endpoint behavior. Flow-Lag therefore lets the policy optimizer use a response-corrected revised multiplier $\widetilde{\lambda}_k$. To determine how the standard multiplier should be corrected, we characterize the local relation between the applied multiplier and the resulting endpoint response. Since the policy update under $\widetilde{\lambda}_k$ produces response S_k, we estimate this local multiplier–response sensitivity from consecutive realized updates as

$$\widehat{h}_k = \frac{S_k - S_{k-1}}{\widetilde{\lambda}_k - \widetilde{\lambda}_{k-1}}. \tag{18}$$

The gain $\widehat{h}_k$ measures whether changing the safety multiplier actually changes the endpoint response. A sufficiently negative gain indicates that additional multiplier pressure remains effective, whereas $|\widehat{h}_k| \approx 0$ indicates a locally saturated multiplier-response relation.

We therefore update the revised multiplier only when the measured response gain remains sufficiently informative:

$$\widetilde{\lambda}_{k+1} = \begin{cases} \left[\lambda_{k+1} - \eta_s \dfrac{S_k + \epsilon}{\widehat{h}_k}\right]_+, & \widehat{h}_k \le -h_{\min}, \\ \widetilde{\lambda}_k + \delta_\lambda, & \widehat{h}_k > -h_{\min}, \end{cases} \tag{19}$$

where $\epsilon > 0$ specifies a target negative endpoint response, $h_{\min} > 0$ is a minimum reliable response gain, and $\delta_\lambda > 0$ is a small multiplier probe introduced to avoid a vanishing denominator in Eq. 18. The derivation and explanation of Eq. 19 are provided in Appendix A.2.

Theorem 1 (Local Stability of Response-Aware Multiplier Adaptation) *Suppose there exists $\lambda^\star > 0$ such that $\mathcal{S}(\lambda^\star) = -\epsilon$. Assume that $\mathcal{S}$ is continuously differentiable near $\lambda^\star$ with $\mathcal{S}'(\lambda^\star) < 0$, and that the safeguarded gain estimate $\widehat{h}_k$ remains negative, bounded, and bounded away from zero in this neighborhood. Then, for a sufficiently small response step size $\eta_s > 0$, there exists $\rho \in (0, 1)$ such that the active response-aware update satisfies*

$$|\widetilde{\lambda}_{k+1} - \lambda^\star| \le \rho|\widetilde{\lambda}_k - \lambda^\star| + |\lambda_{k+1} - \widetilde{\lambda}_k|. \tag{20}$$

Sensitivity-guided update calibration. When the multiplier-response gain remains informative, Flow-Lag adjusts the revised multiplier as described above. If this gain becomes weak, however, further changes in the multiplier are unlikely to produce a meaningful additional endpoint correction. In this case, we suppress the change of revised multiplier.

A weak multiplier response does not necessarily mean that the current policy update is ineffective. If the realized update already decreases endpoint cost, i.e., $S_k < 0$, we instead retain its direction and adjust only how far we move along it. We therefore consider $\alpha > 1$ only when the current direction is cost-decreasing.

For update calibration, we use $\bar{\mathcal{Q}}_C$ as a local cost surrogate along the realized optimizer direction. Around the full candidate, its local variation is approximated as

$$\bar{\mathcal{Q}}_C(\alpha) \approx \bar{\mathcal{Q}}_C(1) + S_k(\alpha - 1), \tag{21}$$

We use this local response to determine the magnitude of an additional cost-reducing extension when the policy remains above the safety budget. This yields the following first-order calibration rule

$$\alpha_k = \min\left\{\alpha_{\max},\ 1 + \beta\frac{\bar{\mathcal{Q}}_C(1) - d}{-S_k}\right\}, \tag{22}$$

applied when $\widehat{h}_k > -h_{\min}$, $\bar{\mathcal{Q}}_C(1) > d$, and $S_k < 0$. Here, $\beta \in (0, 1]$ controls how conservatively the update is extended, and $\alpha_{\max} > 1$ prevents excessively large steps. Otherwise, we use $\alpha_k = 1$. In practice, calibration uses an episodic-scale anchored proxy, with $\bar{\mathcal{Q}}_C$ capturing only the relative cost change along the update direction. The detailed derivation is provided in Appendix A.4.

Integration with Flow RL. Flow-Lag does not alter the internal optimization objective or update procedure of the underlying flow-policy algorithm. Instead, it operates at the primal–dual level. This separation allows the same mechanism to be combined with different flow-based RL backbones. For the main experiments, which train flow policies from scratch, we adopt FPO as the default backbone because it is naturally suited to direct policy optimization from randomly initialized flow policies. For pretrained flow policies, the same response-aware mechanism can instead be applied as a safety fine-tuning module on top of flow-RL methods designed for policy adaptation, such as π_{RL}. We further evaluate this setting in Sec. 5, showing that Flow-Lag can also serve as an effective safety fine-tuning mechanism beyond the FPO setting. The detailed algorithm is listed in Algorithm 1.

5 Experimental Results

We evaluate Flow-Lag on a diverse set of Safety-Gymnasium navigation (Ji et al., 2023) and velocity-constrained Safe MuJoCo benchmarks (Ji et al., 2023). Our experiments are designed to answer five questions: **Q1: Overall performance.** Does Flow-Lag achieve strong performance while satisfying safety constraints compared with other representative safe-RL methods? **Q2: Safety-correction dynamics.** Does endpoint-response feedback produce effective correction of constraint violations for flow optimization? **Q3: Compatibility with different flow-RL backbones.** Can Flow-Lag be integrated with different flow-policy optimization backbones while providing consistent safety improvements? **Q4: Ablation analysis.** What are the individual contributions of response-aware multiplier adaptation and sensitivity-guided update calibration? **Q5: Endpoint sensitivity.** Does the proposed flow sensitivity characterize the changes of the cost of the deployed endpoint policy?

Environments. We evaluate the algorithms on **Safety Gymnasium navigation suite** (Ji et al., 2023) and **Safe MuJoCo velocity-control suite** (Ji et al., 2023), rom the **Safe MuJoCo velocity**

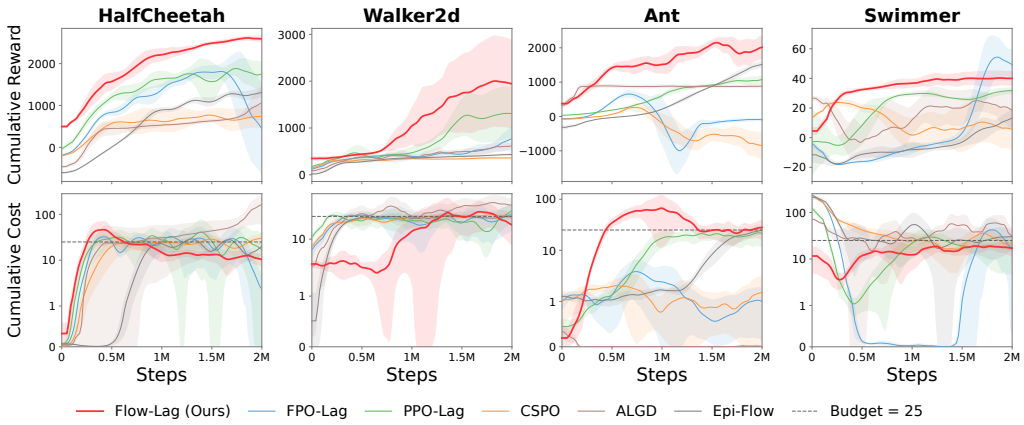

Figure 3: Safe MuJoCo results. Cumulative reward and cost during training.

suite, we consider HalfCheetah, Walker2d, Ant, and Swimmer, which require locomotion reward maximization under velocity-related safety constraints. From the **Safety-Gymnasium navigation suite**, we consider CarButton1, PointButton1, CarGoal1, and PointButton2, covering navigation and goal-directed interaction while avoiding unsafe contacts. In addition, we construct a simple analytically solvable CMDP with a single decision and a two-dimensional continuous action. Its constrained optimum and optimal Lagrange multiplier are available in closed form, allowing us to isolate the multiplier–response behavior and directly verify the proposed endpoint-sensitivity mechanism. Complete environment definitions, the analytic CMDP construction, safety budgets, and training protocols are provided in Appendix B.1.

Baselines. We evaluate two variants of our method with different flow-RL backbones. Unless otherwise specified, Flow-Lag denotes our default implementation built on FPO (McAllister et al., 2026), which supports training flow policies from scratch. We additionally consider **Flow-Lag-π_{RL}**, which integrates the same sensitivity-aware primal–dual mechanism with π_{RL} (Chen et al., 2025). Because this method is designed primarily for adapting pretrained flow policies, we use this variant to evaluate Flow-Lag in the fine-tuning settings. We compare against five representative safe-RL baselines. **FPO-Lag** combines FPO (McAllister et al., 2026) with a standard violation-driven Lagrangian update and therefore provides the most direct flow-policy primal–dual baseline. **PPO-Lag** (Stooke et al., 2020) provides the corresponding classical primal–dual baseline with a Gaussian policy. **CSPO** (Belouadah et al., 2026) incorporates constraint sensitivity into policy correction, but is not designed specifically for flow policies. **ALGD** (Cheng et al., 2026) combines diffusion-based policy optimization with Lagrangian safety guidance, while **Epi-Flow** (Tayal & Tayal, 2026) is an offline flow-based safe-RL method based on an epigraph formulation. Details for all variants and baselines are provided in Appendix B.2.

5.1 Q1: Overall Performance

Figs. 3 and 6 show that Flow-Lag achieves the strongest reward–safety trade-off across both benchmark families. The clearest comparison is with **FPO-Lag**, which shares the same FPO backbone but uses a standard violation-driven dual update. FPO-Lag often exhibits slow cost reduction or overly conservative behavior in Fig 6, consistent with the multiplier–response mismatch we discussed. Compared with **PPO-Lag**, Flow-Lag further benefits from using endpoint response to determine whether a safety correction is actually effective, rather than relying only on constraint violation. **CSPO** also incorporates sensitivity, but uses it primarily to scale a unified constraint correction. Flow-Lag instead uses sensitivity as feedback to decide whether further dual correction is useful and how strongly an effective update should be applied, avoiding both insufficient and excessive correction. **ALGD** constructs a Lagrangian-guided target action distribution and approximates its score using Monte Carlo samples of alternative actions. This introduces an additional distribution-estimation problem, whose accuracy depends on sufficiently covering the relevant action space and can become increasingly challenging in complex control tasks. **Epi-Flow** is trained offline and is therefore limited by the coverage of its fixed dataset.

5.2 Q2: Safety-Correction Dynamics

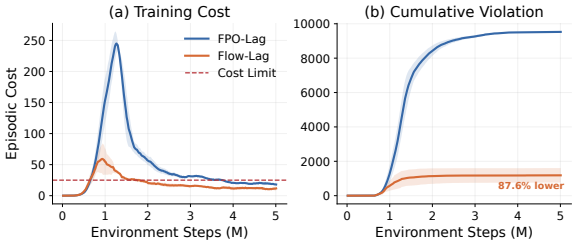

Figure 4: HalfCheetah safety correction.

Figs. 4 and 7 examine how effectively Flow-Lag corrects constraint violations. We measure transient safety failure using cumulative violation, defined as the accumulated excess cost above the safety budget over training. On HalfCheetah, FPO-Lag eventually returns toward the safety budget but only after a large transient violation, whereas Flow-Lag reacts much earlier and reduces cumulative violation by **87.6%**. This indicates that endpoint-response feedback improves not only final constraint satisfaction, but also the timeliness and effectiveness of the safety correction. The same behavior is even clearer in the analytically solvable CMDP. FPO-Lag substantially overshoots the active constraint boundary, while Flow-Lag approaches the target cost with little transient violation, reducing cumulative violation by **99.8%**.

5.3 Q3: Compatibility with different flow-RL backbones

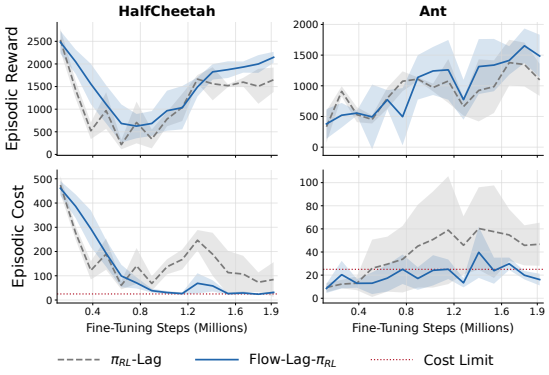

Figure 5: Fine-tuning performance of π_{RL}-Lag and Flow-Lag-π_{RL} on HalfCheetah and Ant.

To examine whether Flow-Lag can be adapted to other flow RL backbones, we further integrate the same sensitivity-aware mechanism with π_{RL} and fine-tune pre-trained flow policies. As shown in Fig. 5, Flow-Lag-π_{RL} consistently improves the reward–safety trade-off over π_{RL}-Lag. On HalfCheetah, both methods recover from the initially unsafe policy, but Flow-Lag-π_{RL} achieves substantially lower cost during later fine-tuning while attaining higher final reward. The benefit is more pronounced on Ant, where π_{RL}-Lag gradually drifts above the safety budget, whereas Flow-Lag-π_{RL} maintains cost close to or below the limit while also achieving higher reward. These results show that the proposed response-aware correction is not specific to FPO and can be integrated with different flow-policy optimizers to improve safety during fine-tuning.

Q4. Ablation Analysis Fig. 8 isolates the two components of Flow-Lag. **Response Only** retains the response-aware multiplier adaptation but fixes the optimizer step scale $\alpha = 1$. **Calibration Only** uses the standard multiplier update and applies only the sensitivity-guided step calibration α. The full method combines both mechanisms. The results show that the two components serve complementary roles. When the multiplier–response sensitivity is informative, Flow-Lag uses it to adapt λ and obtain an effective cost-reducing update. When this response becomes weak, further increasing λ provides little additional safety correction; Flow-Lag instead suppresses multiplier escalation and calibrates α to exploit an already cost-decreasing update. This explains why **Calibration Only** performs poorly in safety: step calibration can control how far the current update is taken, but cannot make an ineffective or poorly directed update become cost-decreasing. Without response-aware multiplier adaptation, the underlying dual update can therefore remain weak or saturated, leading to substantially higher episodic cost and violation rates. Conversely, **Response Only** provides much stronger safety control, but cannot exploit additional progress along an effective update once the multiplier channel becomes insensitive.

6 Conclusion

In this work, we identified an optimization mismatch problem in primal–dual flow RL, where stronger safety penalties do not necessarily produce effective endpoint cost reduction. To address this, we proposed Flow-Lag, which uses sensitivity to measure the safety response of the actual flow policy update and accordingly calibrate multiplier adaptation and update magnitude. Experiments on Safe MuJoCo and Safety Gymnasium show that Flow-Lag improves reward–safety performance, reduces transient constraint violations, and provides reliable sensitivity estimates.

AI USE STATEMENT

Generative AI tools were used to assist with language polishing and to support code generation during the development of this work. All AI-assisted code was manually reviewed, tested, and validated by the authors before being used in the experiments. The authors take full responsibility for the correctness of the manuscript, implementation, and reported results.

ETHICS STATEMENT

This work studies safe reinforcement learning for flow-based policies, with the goal of improving constraint satisfaction during policy optimization. All experiments are conducted entirely in simulated environments and do not involve human participants, personal data, or deployment on physical systems. We therefore do not identify any direct ethical concerns arising from the experimental procedures used in this work.

REFERENCES

Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. Constrained policy optimization. In *International conference on machine learning*, pp. 22–31. Pmlr, 2017.

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.

Ayoub Belouadah, Sylvain Kubler, and Yves Le Traon. Cspo: Constraint-sensitive policy optimization for safe reinforcement learning. *arXiv preprint arXiv:2606.14415*, 2026.

Kang Chen, Zhihao Liu, Tonghe Zhang, Zhen Guo, Si Xu, Hao Lin, Hongzhi Zang, Quanlu Zhang, Zhaofei Yu, Guoliang Fan, et al. πrl: Online rl fine-tuning for flow-based vision-language-action models. *arXiv preprint arXiv:2510.25889*, 8, 2025.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Weiqin Chen, James Onyejizu, Long Vu, Lan Hoang, Dharmashankar Subramanian, Koushik Kar, Sandipan Mishra, and Santiago Paternain. Adaptive primal-dual method for safe reinforcement learning. *arXiv preprint arXiv:2402.00355*, 2024.

Xiaoyuan Cheng, Wenxuan Yuan, Boyang Li, Yuanchao Xu, Yiming Yang, Hao Liang, Bei Peng, Robert Loftin, Zhuo Sun, and Yukun Hu. How does the lagrangian guide safe reinforcement learning through diffusion models? *arXiv preprint arXiv:2602.02924*, 2026.

Yinlam Chow, Ofir Nachum, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31, 2018.

Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. Safe exploration in continuous action spaces. *arXiv preprint arXiv:1801.08757*, 2018.

Quan Dao, Hao Phung, Binh Nguyen, and Anh Tran. Flow matching in latent space. *arXiv preprint arXiv:2307.08698*, 2023.

Dongsheng Ding, Kaiqing Zhang, Tamer Basar, and Mihailo Jovanovic. Natural policy gradient primal-dual method for constrained markov decision processes. *Advances in Neural Information Processing Systems*, 33:8378–8390, 2020.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Jonathan Ho, Xi Chen, Aravind Srinivas, Yan Duan, and Pieter Abbeel. Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *International conference on machine learning*, pp. 2722–2730. PMLR, 2019.

Jiaming Ji, Borong Zhang, Jiayi Zhou, Xuehai Pan, Weidong Huang, Ruiyang Sun, Yiran Geng, Yifan Zhong, Josef Dai, and Yaodong Yang. Safety gymnasium: A unified safe reinforcement learning benchmark. *Advances in Neural Information Processing Systems*, 36:18964–18993, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

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

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-grpo: Training flow matching models via online rl. *Advances in neural information processing systems*, 38:40783–40818, 2026.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Olvi L Mangasarian. *Nonlinear programming*. SIAM, 1994.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *International Conference on Learning Representations*, volume 2026, pp. 36352–36372, 2026.

Santiago Paternain, Luiz Chamon, Miguel Calvo-Fullana, and Alejandro Ribeiro. Constrained reinforcement learning has zero duality gap. *Advances in Neural Information Processing Systems*, 32, 2019.

Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7(1):2, 2019.

Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

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

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by pid lagrangian methods. In *International conference on machine learning*, pp. 9133–9143. PMLR, 2020.

Manan Tayal and Mumuksh Tayal. Epigraph-guided flow matching for safe and performant offline reinforcement learning. *arXiv preprint arXiv:2602.08054*, 2026.

Mumuksh Tayal, Manan Tayal, and Ravi Prakash. Safe flow q-learning: Offline safe reinforcement learning with reachability-based flow policies. *arXiv preprint arXiv:2603.15136*, 2026.

Chen Tessler, Daniel J Mankowitz, and Shie Mannor. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074*, 2018.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *arXiv preprint arXiv:2302.00482*, 2023.

Zifan Wu, Bo Tang, Qian Lin, Chao Yu, Shangqin Mao, Qianlong Xie, Xingxing Wang, and Dong Wang. Off-policy primal-dual safe reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 11371–11390, 2024.

Tengyu Xu, Yingbin Liang, and Guanghui Lan. Crpo: A new approach for safe reinforcement learning with convergence guarantee. In *International Conference on Machine Learning*, pp. 11480–11491. PMLR, 2021.

Jeongyong Yang, Seunghwan Jang, and SooJean Han. Safeflowmatcher: Safe and fast planning using flow matching with control barrier functions. In *International Conference on Learning Representations*, volume 2026, pp. 54058–54087, 2026.

Long Yang, Jiaming Ji, Juntao Dai, Linrui Zhang, Binbin Zhou, Pengfei Li, Yaodong Yang, and Gang Pan. Constrained update projection approach to safe policy optimization. *Advances in Neural Information Processing Systems*, 35:9111–9124, 2022.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *Advances in Neural Information Processing Systems*, 38: 106282–106319, 2026.

A Mathematical Proofs

A.1 Proof of Lemma 1: Endpoint-Cost Decrease

Proof 1 *Define the update path*

$$\theta_k(\alpha) = \theta_k + \alpha\Delta\theta_k, \qquad \alpha \in [0,1],$$

where $\Delta\theta_k = \theta_k^+ - \theta_k$. During this local sensitivity evaluation, the sampled batch and cost critic are held fixed.

By definition

$$S_k = \left.\frac{d\,\bar{\mathcal{Q}}_C(\theta_k(\alpha))}{d\alpha}\right|_{\alpha=1} = \nabla_\theta \bar{\mathcal{Q}}_C(\theta_k^+)^\top \Delta\theta_k.$$

Applying the chain rule through the K flow steps gives

$$S_k = \mathbb{E}_{(s,z)}\left[\sum_{j=0}^{K-1} (g_{k,j}^C)^\top \delta v_{k,j}\right],$$

where $g_{k,j}^C$ is the endpoint cost sensitivity to the jth flow velocity and

$$\delta v_{k,j} = \left.\frac{\partial v_\theta(x_j, \sigma_j; s)}{\partial\theta}\right|_{\theta=\theta_k^+} \Delta\theta_k.$$

Since $\bar{\mathcal{Q}}_C$ has an L_C-Lipschitz continuous gradient, the first-order Taylor remainder around θ_k^+ satisfies

$$\left|\bar{\mathcal{Q}}_C(\theta_k) - \bar{\mathcal{Q}}_C(\theta_k^+) + \nabla_\theta \bar{\mathcal{Q}}_C(\theta_k^+)^\top \Delta\theta_k\right| \le \frac{L_C}{2}\|\Delta\theta_k\|^2.$$

Therefore

$$\bar{\mathcal{Q}}_C(\theta_k^+) - \bar{\mathcal{Q}}_C(\theta_k) \le S_k + \frac{L_C}{2}\|\Delta\theta_k\|^2,$$

which proves Eq. 16.

Hence, whenever

$$S_k < -\frac{L_C}{2}\|\Delta\theta_k\|^2,$$

we obtain

$$\bar{\mathcal{Q}}_C(\theta_k^+) < \bar{\mathcal{Q}}_C(\theta_k).$$

A.2 Derivation of the Response-Aware Multiplier Update

We are motivated from Newton-Raphson method and derive the response-aware multiplier update. Recall that S_k denotes the endpoint response produced by the policy update under the current revised multiplier $\widetilde{\lambda}_k$. We seek an revised multiplier whose endpoint response attains a small negative target

$$S(\widetilde{\lambda}) = -\epsilon, \qquad \epsilon > 0, \tag{23}$$

where the negative target maintains a local cost-decreasing response.

Define the response error

$$F(\widetilde{\lambda}) \triangleq S(\widetilde{\lambda}) + \epsilon. \tag{24}$$

The desired multiplier therefore corresponds locally to a root of $F(\widetilde{\lambda})$.

Because the mapping from the applied multiplier to the realized endpoint response is mediated by the complete flow-policy optimizer, its derivative is not available in closed form. We instead use the measured response gain

$$\widehat{h}_k = \frac{S_k - S_{k-1}}{\widetilde{\lambda}_k - \widetilde{\lambda}_{k-1}} \tag{25}$$

as a local secant estimate of the response variation. Around $\widetilde{\lambda}_k$, the response can then be locally approximated as

$$S(\lambda) \approx S_k + \widehat{h}_k(\lambda - \widetilde{\lambda}_k).$$

Imposing the target condition in Eq. 23 on this local model gives

$$-\epsilon = S_k + \widehat{h}_k(\lambda - \widetilde{\lambda}_k),$$

and hence

$$\lambda = \widetilde{\lambda}_k - \frac{S_k + \epsilon}{\widehat{h}_k}.$$

This is the corresponding local Newton-Raphson root-finding correction, with $\widehat{h}_k$ replacing the unavailable exact response slope. Introducing a damping coefficient $\eta_s > 0$, the local response correction is

$$-\eta_s \frac{S_k + \epsilon}{\widehat{h}_k}.$$

Flow-Lag does not replace the standard violation-driven primal-dual update with this correction. Instead, it preserves the standard multiplier λ_{k+1} and adjusts it according to the realized endpoint response. Equivalently, starting from the current revised multiplier, the response correction is combined with the standard primal-dual change:

$$\begin{aligned}\widetilde{\lambda}_{k+1} &= \left(\widetilde{\lambda}_k - \eta_s \frac{S_k + \epsilon}{\widehat{h}_k}\right) + \left(\lambda_{k+1} - \widetilde{\lambda}_k\right) \\ &= \lambda_{k+1} - \eta_s \frac{S_k + \epsilon}{\widehat{h}_k}.\end{aligned}$$

Thus, the update can be interpreted as the standard primal-dual multiplier augmented by a damped local root-finding correction that regulates the realized endpoint response toward $-\epsilon$.

The sign of this correction provides the desired feedback when $\widehat{h}_k < 0$. If $S_k > -\epsilon$, then $S_k + \epsilon > 0$, so the correction is positive and increases the revised multiplier, strengthening the cost-decreasing response. Conversely, if $S_k < -\epsilon$, the correction is negative and reduces the revised multiplier, avoiding unnecessarily aggressive constraint penalty.

Finally, the correction is applied only when $\widehat{h}_k \le -h_{\min}$. When the response gain is close to zero, the multiplier-response channel is locally weak, and division by $\widehat{h}_k$ can produce an unreliable or excessively large correction. Flow-Lag therefore suppresses the response correction in this regime and suppresses the previous revised multiplier, yielding

$$\widetilde{\lambda}_{k+1} = \begin{cases} \left[\lambda_{k+1} - \eta_s \dfrac{S_k + \epsilon}{\widehat{h}_k}\right]_+, & \widehat{h}_k \le -h_{\min}, \\ \widetilde{\lambda}_k + \delta_\lambda, & \widehat{h}_k > -h_{\min}, \end{cases} \tag{26}$$

A.3 Proof of Theorem 1

Proof 2 *Since $\mathcal{S}(\lambda^\star) + \epsilon = 0$, the mean-value theorem gives, for some ξ_k between $\widetilde{\lambda}_k$ and $\lambda^\star$,*

$$\mathcal{S}(\widetilde{\lambda}_k) + \epsilon = \mathcal{S}'(\xi_k)(\widetilde{\lambda}_k - \lambda^\star).$$

Substituting this relation into the active branch of Eq. 19 yields

$$\begin{aligned}\widetilde{\lambda}_{k+1} - \lambda^\star &= \lambda_{k+1} - \lambda^\star - \eta_s \frac{\mathcal{S}'(\xi_k)}{\widehat{h}_k}(\widetilde{\lambda}_k - \lambda^\star) \\ &= \left(1 - \eta_s \frac{\mathcal{S}'(\xi_k)}{\widehat{h}_k}\right)(\widetilde{\lambda}_k - \lambda^\star) + (\lambda_{k+1} - \widetilde{\lambda}_k).\end{aligned} \tag{27}$$

By continuity of $\mathcal{S}'$ and the assumed safeguards on $\widehat{h}_k$, the ratio $\mathcal{S}'(\xi_k)/\widehat{h}_k$ is positive and remains locally bounded away from both zero and infinity. Therefore, for sufficiently small $\eta_s > 0$, there exists $\rho \in (0, 1)$ such that

$$\left|1 - \eta_s \frac{\mathcal{S}'(\xi_k)}{\widehat{h}_k}\right| \le \rho.$$

Taking absolute values in Eq. 27 gives

$$|\widetilde{\lambda}_{k+1} - \lambda^\star| \le \rho|\widetilde{\lambda}_k - \lambda^\star| + |\lambda_{k+1} - \widetilde{\lambda}_k|,$$

which proves Eq. 20.

Finally, repeated application of the contraction shows that if $\lambda_{k+1} - \widetilde{\lambda}_k \to 0$, then $\widetilde{\lambda}_k \to \lambda^\star$.

A.4 Derivation of the Calibration Function

Let

$$\bar{\mathcal{Q}}_C(\alpha) \triangleq \bar{\mathcal{Q}}_C\big(\theta_k + \alpha\Delta\theta_k\big),$$

where $\alpha = 1$ corresponds to the full optimizer candidate. Recall that

$$S_k = \bar{\mathcal{Q}}'_C(1).$$

Using a first-order approximation around the full candidate gives

$$\bar{\mathcal{Q}}_C(\alpha) \approx \bar{\mathcal{Q}}_C(1) + S_k(\alpha - 1).$$

Suppose that the full candidate remains above the safety budget, $\bar{\mathcal{Q}}_C(1) > d$, and that the realized optimizer direction is locally cost-decreasing, $S_k < 0$. We seek an extension $\alpha > 1$ whose first-order predicted endpoint cost reaches the budget d. Setting the linear model in Eq. 21 equal to d gives

$$d = \bar{\mathcal{Q}}_C(1) + S_k(\alpha - 1).$$

Solving for α yields

$$\begin{aligned}\alpha - 1 &= \frac{d - \bar{\mathcal{Q}}_C(1)}{S_k} \\ &= \frac{\bar{\mathcal{Q}}_C(1) - d}{-S_k},\end{aligned}$$

and therefore the full first-order correction is

$$\alpha^\star = 1 + \frac{\bar{\mathcal{Q}}_C(1) - d}{-S_k}.$$

Because $\bar{\mathcal{Q}}_C(1) > d$ and $S_k < 0$, the correction in Eq. 21 is positive, so $\alpha^\star > 1$.

Since Eq. 28 is only a local first-order approximation, Flow-Lag applies only a fraction $\beta \in (0, 1]$ of the predicted extension. This gives

$$\alpha_k^{\text{trial}} = 1 + \beta\frac{\bar{\mathcal{Q}}_C(1) - d}{-S_k}.$$

Under the local linear model, this choice reduces the predicted constraint gap by a fraction β, since

$$\begin{aligned}\bar{\mathcal{Q}}_C\big(\alpha_k^{\text{trial}}\big) - d &\approx \bar{\mathcal{Q}}_C(1) - d + S_k\beta\frac{\bar{\mathcal{Q}}_C(1) - d}{-S_k} \\ &= (1 - \beta)\left(\bar{\mathcal{Q}}_C(1) - d\right).\end{aligned}$$

Thus, $\beta = 1$ targets the safety budget under the local linear model, whereas $\beta < 1$ provides a conservative partial correction.

Finally, to prevent excessively large extensions when $|S_k|$ is small or when the local linear approximation is inaccurate, we impose the maximum scale $\alpha_{\max} > 1$, yielding

$$\alpha_k = \min\left\{\alpha_{\max},\ 1 + \beta\frac{\bar{\mathcal{Q}}_{C,k}(1) - d}{-S_k}\right\}.$$

A.5 Proof of Proposition 1

Proposition 1 (Local Joint Convergence of Response-Aware Primal–Dual Dynamics) *Let*

$$\mathcal{T}_{\mathcal{B}}(\theta, \widetilde{\lambda})$$

denote the deterministic mean candidate produced by one complete update of the underlying flow-RL backbone $\mathcal{B}$ using revised multiplier $\widetilde{\lambda}$, and define the corresponding mean policy displacement as

$$U(\theta, \widetilde{\lambda}) := \mathcal{T}_{\mathcal{B}}(\theta, \widetilde{\lambda}) - \theta.$$

Thus, in the informative-response regime, the mean primal update is $\theta_{k+1} = \theta_k + U(\theta_k, \widetilde{\lambda}_k)$.

Suppose $(\theta^\star, \lambda^\star)$ is an active KKT point. Assume that the corresponding standard primal–dual mean dynamics are locally exponentially stable around $(\theta^\star, \lambda^\star)$, and that U, S, and h are continuously differentiable in a neighborhood where the informative-response branch is active and the multiplier projection is inactive, with

$$U(\theta^\star, \lambda^\star) = 0, \qquad S(\theta^\star, \lambda^\star) = 0, \qquad h^\star := h(\theta^\star, \lambda^\star) < -h_{\min}.$$

Then, for any fixed $\epsilon > 0$, there exists $\bar{\eta}_s > 0$ such that, for every $0 < \eta_s < \bar{\eta}_s$, the coupled Flow-Lag mean dynamics are locally exponentially stable around

$$z_\epsilon^\star = \left(\theta^\star,\, \lambda^\star + \eta_s \frac{\epsilon}{h^\star},\, \lambda^\star\right).$$

where the second and third components denote the standard and revised multipliers, respectively. Consequently,

$$\theta_k \to \theta^\star, \qquad \widetilde{\lambda}_k \to \lambda^\star, \qquad \lambda_k \to \lambda^\star + \eta_s \frac{\epsilon}{h^\star}.$$

Thus, a fixed response margin changes only the steady-state offset between the standard and revised multipliers, while preserving the limiting active KKT policy and revised multiplier.

Proof 3 *Let*

$$\xi(\theta) := \frac{J_C(\theta) - d}{d}, \qquad r(\theta, \widetilde{\lambda}) := -\frac{S(\theta, \widetilde{\lambda}) + \epsilon}{h(\theta, \widetilde{\lambda})}.$$

In the informative-response regime, the mean Flow-Lag dynamics are

$$\theta^+ = \theta + U(\theta, \widetilde{\lambda}), \qquad \lambda^+ = \lambda + \eta_\lambda \xi(\theta), \qquad \widetilde{\lambda}^+ = \lambda^+ + \eta_s r(\theta, \widetilde{\lambda}).$$

Consider

$$z_\epsilon^\star = (\theta^\star, \lambda^\star_{\text{standard}}, \lambda^\star), \qquad \lambda^\star_{\text{standard}} = \lambda^\star + \eta_s \frac{\epsilon}{h^\star}.$$

At the active KKT point,

$$U(\theta^\star, \lambda^\star) = 0, \qquad \xi(\theta^\star) = 0, \qquad S(\theta^\star, \lambda^\star) = 0.$$

Hence

$$\theta^+ = \theta^\star, \qquad \lambda^+ = \lambda^\star_{\text{standard}},$$

and

$$\widetilde{\lambda}^+ = \lambda^\star_{\text{standard}} - \eta_s \frac{\epsilon}{h^\star} = \lambda^\star.$$

Thus $z_\epsilon^\star$ is a fixed point. It remains to establish local stability. When $\eta_s = 0$, $\widetilde{\lambda}^+ = \lambda^+$. Defining $e := \widetilde{\lambda} - \lambda$ therefore gives $e^+ = 0$. Consequently, the joint linearization has the eigenvalues of the underlying standard primal–dual linearization together with one additional eigenvalue at zero. By assumption, the former is Schur stable, and hence the full Jacobian at $\eta_s = 0$ is also Schur stable.

For $\eta_s > 0$, the response-aware term is a continuously differentiable perturbation of this stable system. Since $|h^\star| \geq h_{\min} > 0$, its Jacobian remains finite, and the joint Jacobian satisfies

$$M_{\eta_s} = M_0 + \mathcal{O}(\eta_s).$$

Table 1: Safe MuJoCo velocity-control environments used in our experiments.

Agent	Environment ID
HalfCheetah	`SafetyHalfCheetahVelocity-v0`
Walker2d	`SafetyWalker2dVelocity-v0`
Ant	`SafetyAntVelocity-v1`
Swimmer	`SafetySwimmerVelocity-v1`

Table 2: Safety-Gymnasium navigation environments used in our experiments.

Task	Environment ID
CarButton1	`SafetyCarButton1-v0`
PointButton1	`SafetyPointButton1-v0`
CarGoal1	`SafetyCarGoal1-v0`
PointButton2	`SafetyPointButton2-v0`

By continuity of the eigenvalues with respect to the matrix entries, there exists $\bar{\eta}_s > 0$ such that

$$\rho(M_{\eta_s}) < 1, \qquad 0 < \eta_s < \bar{\eta}_s.$$

Therefore $z_\epsilon^\star$ is locally exponentially stable.

It follows that

$$\theta_k \to \theta^\star, \qquad \widetilde{\lambda}_k \to \lambda^\star, \qquad \lambda_k \to \lambda^\star + \eta_s \frac{\epsilon}{h^\star}.$$

Since $h^\star < 0$,

$$\widetilde{\lambda}^\star - \lambda^\star_{\mathrm{nom}} = \frac{\eta_s \epsilon}{|h^\star|}.$$

Thus the fixed response margin induces only a steady-state offset between the nominal and effective multipliers, while preserving the active KKT policy and effective multiplier.

B Experimental Setup

We evaluate Flow-Lag on eight continuous-control safety tasks from two complementary benchmark families: four velocity-constrained locomotion tasks from Safe MuJoCo and four navigation tasks from Safety-Gymnasium. Unless otherwise stated, all online methods are trained with five random seeds, $\{111, 112, 113, 114, 115\}$, and use the same episodic safety budget $d = 25$. We additionally use an analytically solvable CMDP to isolate and validate the proposed endpoint-response mechanism.

B.1 Benchmark Environments

Safe MuJoCo velocity-control tasks. We consider four locomotion environments: **HalfCheetah**, **Walker2d**, **Ant**, and **Swimmer**. The agent receives the standard task reward associated with forward locomotion and control, while safety is imposed through a velocity-based constraint. The episodic cost accumulates violations of the prescribed velocity limit, creating a direct trade-off between fast locomotion and safe behavior. The exact environment IDs are listed in Table 1.

Safety-Gymnasium navigation tasks. We further consider four navigation environments: **CarButton1**, **PointButton1**, **CarGoal1**, and **PointButton2**. These tasks cover two robot embodiments, *Car* and *Point*, and two task families, *Goal* and *Button*. Goal tasks reward progress toward and successful arrival at a target, whereas Button tasks require the agent to navigate to and interact with a designated button. Cost is generated by environment-defined unsafe interactions, such as contacts with hazards or incorrect objects. Thus, these tasks require the agent to maximize task completion reward while avoiding unsafe contacts during navigation.

Diagnostic analytic CMDP. In addition to the benchmark environments, we construct a minimal analytically solvable CMDP to study multiplier response and endpoint sensitivity in isolation. The problem consists of a single decision with two-dimensional action $a = (a_1, a_2) \in \mathbb{R}^2$, with

$$r(a) = -\frac{1}{2}\left[(a_1 - 1)^2 + (a_2 - 1)^2\right], \qquad c(a) = a_1^2, \tag{28}$$

and safety budget $d = 0.25$. The reward favors $a = (1, 1)$, whereas the safety constraint requires $|a_1| \leq 0.5$. The constrained optimum is therefore available in closed form:

$$a^\star = (0.5, 1), \qquad J_C^\star = 0.25, \qquad J_R^\star = -0.125, \qquad \lambda^\star = 0.5. \tag{29}$$

The policy transports $z \sim \mathcal{N}(0, I)$ through eight explicit Euler steps,

$$x_{j+1} = x_j + \frac{1}{8} v_j(x_j), \qquad v_j(x) = w_j \odot x + b_j, \tag{30}$$

using learnable diagonal affine velocity fields. We initialize $w_j = (-2, -2)$ and $b_j = (0.45, 0)$, producing a feasible but reward-suboptimal initial policy.

Because this environment removes long-horizon credit assignment, state-distribution shift, and critic approximation, its exact solution provides a controlled reference for evaluating multiplier response, endpoint sensitivity, and safety-correction dynamics.

B.2 Methods and Baselines

We evaluate two instantiations of our method with different flow RL backbones. The default **Flow-Lag** uses FPO as the flow-policy optimization backbone and is used for all from-scratch experiments. We additionally consider **Flow-Lag-π_{RL}**, which integrates the same response-aware primal–dual mechanism with π_{RL} for fine-tuning pretrained flow policies. For the main from-scratch experiments, we compare against five representative safe-RL baselines: FPO-Lag, PPO-Lag, CSPO, ALGD, and Epi-Flow. For the fine-tuning experiment, Flow-Lag-π_{RL} is compared against the corresponding π_{RL}-Lag baseline.

Flow-Lag-π_{RL} (Ours). To evaluate whether the proposed mechanism generalizes beyond FPO, we also integrate Flow-Lag with π_{RL} ODE version. Unlike the default Flow-Lag setting, which trains a flow policy from scratch, this variant is used to fine-tune a pretrained stochastic flow policy. The underlying π_{RL} optimizer retains its path-space PPO-style update, while the same endpoint-response signal is used to adapt the revised multiplier and calibrate the realized update. We compare this variant with π_{RL}-Lag, which applies a standard violation-driven Lagrangian update to the same π_{RL} backbone.

FPO-Lag. FPO-Lag (Schulman et al., 2017) is the closest baseline to our default method. It uses the same FPO backbone as Flow-Lag, but combines reward and cost advantages using a standard violation-driven Lagrange multiplier without endpoint-response feedback. FPO constructs a PPO-style surrogate from changes in the conditional-flow-matching objective rather than from an explicit endpoint likelihood.

We use 8 flow integration steps and 8 Monte Carlo samples for the conditional-flow-matching surrogate. The actor uses hidden width 256 and learning rate 3×10^{-4}, while the reward and cost critics use learning rate 10^{-3}. We use discount factor $\gamma = 0.99$ and GAE (Schulman et al., 2015) parameter $\lambda_{\mathrm{GAE}} = 0.95$.

PPO-Lag. PPO-Lag is a conventional primal–dual baseline with a Gaussian actor. The actor is optimized using the PPO clipped surrogate, with reward and cost advantages combined through a dynamically updated Lagrange multiplier. The multiplier increases when mean episodic cost exceeds the safety budget and decreases otherwise. We use a PPO clipping coefficient of 0.2, while the remaining shared on-policy optimization settings are aligned with the flow-policy baselines.

CSPO. We use a local expected-cost implementation of a CSPO-style sensitivity correction. The method augments the policy objective with a sensitivity-weighted quadratic constraint correction,

$$L_{\mathrm{actor}} = -L_R + s(t)\left[\lambda g(\theta) + \frac{\alpha}{2} w [g(\theta)]_+^2\right], \tag{31}$$

where $g(\theta)$ denotes the expected-cost constraint residual and

$$w \approx \frac{1}{\|\nabla g\|^2 + \epsilon}. \tag{32}$$

The sensitivity weight is clipped and stabilized with an exponential moving average.

We set $\alpha = 0.3$. Training begins with a 100k-step reward-only warm start followed by a 100k-step linear safety ramp. For Safe MuJoCo, the actor uses a full-batch update with minibatch size 4096. We therefore refer to this baseline as our local expected-cost CSPO implementation rather than an unmodified external implementation.

ALGD. ALGD is a diffusion-policy safe-RL baseline that combines a diffusion actor with reward and cost critics and constructs a Lagrangian-guided target action distribution. Its target score is estimated from multiple sampled alternative actions, introducing an additional Monte Carlo distribution-estimation step. We use the vendored official ALGD implementation and retain its network architecture, critic structure, and diffusion-guidance objective. The policy uses 5 diffusion steps, 4 score Monte Carlo samples, and a cost-critic ensemble of size 4.

Our ALGD experiments use a compute-reduced optimization schedule. Each seed collects the same environment-interaction budget as the other online methods, but performs one replay-gradient update every 2 environment steps. The first 5,000 environment steps use random exploration, with batch size 256 and replay capacity 10^6.

Epi-Flow. Epi-Flow is an offline flow-based safe-RL baseline based on an epigraph formulation. It jointly learns reward, safety, and auxiliary epigraph value functions together with a conditional flow policy from a fixed offline dataset. During deployment, the method searches for a feasible epigraph variable, samples candidate flow actions, and selects the candidate with the largest auxiliary safety value.

For each environment, we construct one shared dataset containing 10^6 transitions collected from a mixture of FPO-Lag behavior policies and random actions. The five training seeds use the same dataset but independently initialize all learned components. Training consists of 50k value-function updates and 50k flow-policy updates, with batch size 256, discount factor 0.99, expectile parameter 0.9, target-network EMA coefficient 0.005, and 5 flow steps.

For Safety-Gymnasium tasks, we construct a transition-level binary safety label from the environment-reported cost: transitions with zero cost receive a positive safety label, while transitions with nonzero cost receive a negative label. This transition-level objective differs from the expected episodic-cost constraint used by the online methods. Accordingly, Epi-Flow should be interpreted as a complementary offline reference rather than a strictly matched online baseline.

The CSPO implementation does not use all of these actor-update settings: for Safe MuJoCo it uses a full-batch actor update with batch size 4096, together with the 100k reward-only warm start and 100k linear safety ramp described above. ALGD follow its respective implementation-specific optimization procedures.

C Additional Experimental Results

C.1 Q5. Endpoint sensitivity

Fig. 9 evaluates whether the proposed flow sensitivity correctly captures the endpoint cost response of the realized policy update. In the analytic CMDP, the estimated sensitivity matches the exact derivative to numerical precision, achieving nearly **100% sign agreement**. On SafetyAntVelocity, where no closed-form derivative is available, we compare against finite-difference estimates and obtain **96.7% sign agreement** with a median absolute error of **0.034**. Here, sign agreement measures the fraction of policy updates for which the estimated and reference sensitivities have the same sign. A negative sign indicates that the realized update locally decreases endpoint cost, whereas a positive sign indicates the opposite. This directional information is particularly important for Flow-Lag, since it determines whether a safety correction is effective and whether multiplier adaptation or update calibration should be applied. The high agreement in both settings therefore confirms that the proposed sensitivity provides a reliable signal for response-aware safety control.

Table 3: Method-specific hyperparameters and implementation details for all baselines.

Method	Hyperparameter	Value
FPO-Lag	Flow integration steps	8
	Flow-matching Monte Carlo samples	8
	Policy clipping coefficient	0.2
π_{RL}-Lag	Path-ratio formulation	State-conditioned, non-joint
	Stochastic flow steps	8
	Flow noise standard deviation	0.5
	Policy clipping coefficient	0.2
PPO-Lag	Policy clipping coefficient	0.2
CSPO	Quadratic correction coefficient α_{CSPO}	0.3
	Reward-only warm-up	100,000 environment steps
	Safety-ramp duration	100,000 environment steps
	Safety-ramp schedule	Linear
	Actor minibatch size (Safe MuJoCo)	4096 (full batch)
	Sensitivity stabilization	Gradient-based weighting, clipping, and EMA
ALGD	Implementation	Vendored official implementation
	Diffusion steps	5
	Score Monte Carlo samples	4
	Cost-critic ensemble size	4
	Random-exploration steps	5,000
	Replay batch size / capacity	256 / 10^6
	Replay-gradient frequency	One update every 2 environment steps
Epi-Flow	Training protocol	Offline
	Offline dataset size	10^6 transitions per environment
	Value-function updates	50,000
	Flow-policy updates	50,000
	Batch size	256
	Discount factor γ	0.99
	Expectile parameter	0.9
	Target-network EMA coefficient	0.005
	Flow steps / candidate actions	5 / 8

D FLOW-LAG ALGORITHM

Algorithm 1 summarizes the complete Flow-Lag procedure. Flow-Lag treats the underlying flow RL optimizer as a modular backbone: in our from-scratch experiments we use FPO, while in the fine-tuning experiments we replace it with π_{RL}. The response-aware multiplier adaptation and update calibration are otherwise unchanged.

E FLOW-LAG HYPERPARAMETERS

Table 4 reports the main optimization and response-controller hyperparameters used by Flow-Lag. The endpoint response is estimated using a fixed cost critic on a calibration batch sampled from the current rollout.

The navigation dual learning rate is task dependent: we use 0.01 for the Button tasks and up to 0.035 for the Goal tasks. All other controller parameters are shared across the navigation tasks.

The navigation dual learning rate is task dependent: we use 0.01 for the Button tasks and up to 0.035 for the Goal tasks. All other controller parameters are shared across the navigation tasks. Observation-normalization statistics are frozen while collecting and optimizing each rollout.

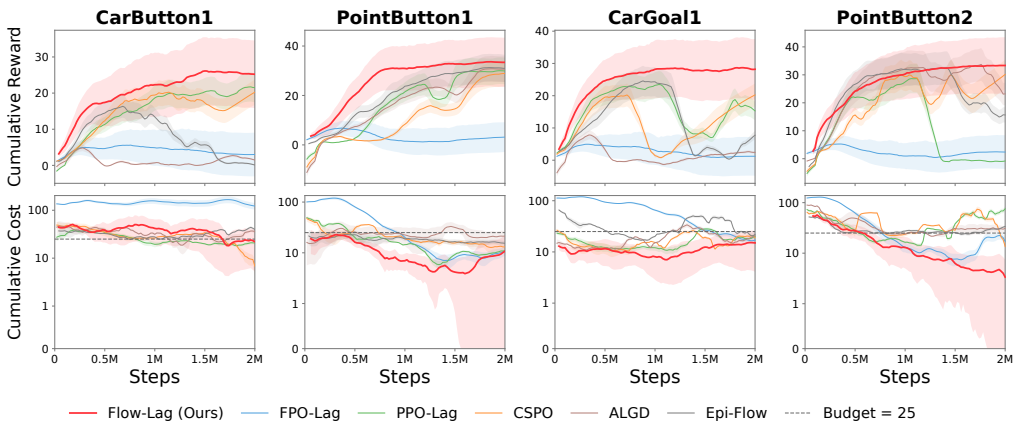

Figure 6: Safety-Gymnasium results. Cumulative reward and cost on navigation tasks.

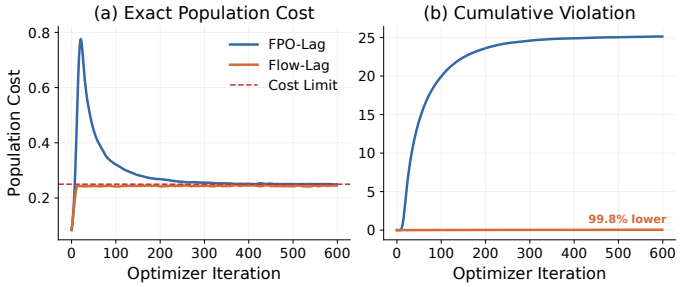

Figure 7: **Analytic CMDP.** Flow-Lag reduces cumulative violation by 99.8% while converging near the active constraint boundary.

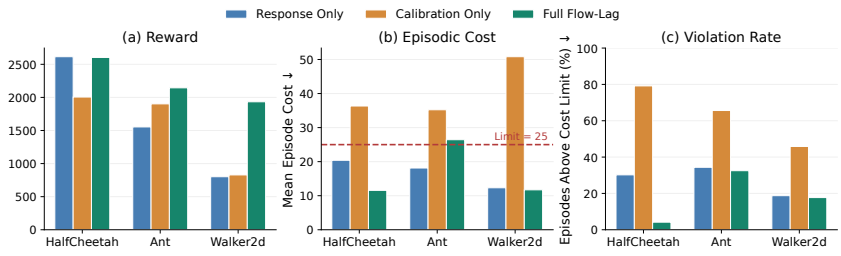

Figure 8: Component ablation.

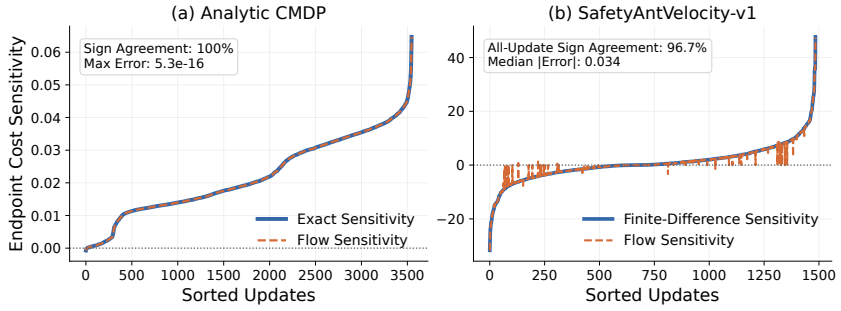

Figure 9: Endpoint sensitivity validation.

Algorithm 1 Flow-Lag with a Generic Flow-RL Backbone

Require: Flow-policy parameters θ; reward value critic V_R; cost value critic V_C; endpoint cost critic Q_C; cost budget d; standard multiplier λ; revised multiplier $\tilde{\lambda}$; dual step size η_λ; response step size η_s; target response ϵ; response threshold $h_{\min}$; multiplier probe δ_λ; calibration parameters $\beta, \alpha_{\max}$; flow-RL backbone $\mathcal{B} \in \{\mathrm{FPO}, \pi_{RL}\}$.

for $k = 0, 1, 2, \ldots$ **do**
  Collect trajectories using the current flow policy π_{θ_k}.
  Update V_R, V_C, and Q_C, and estimate reward and cost advantages $\widehat{A}_{R,k}$ and $\widehat{A}_{C,k}$.
  Compute the mean episodic cost $\bar{J}_{C,k}$.
  Update the standard multiplier λ_{k+1} using Eq. 17.
  Flow-policy optimization backbone:

$$\theta_k^+ \leftarrow \textsc{FlowUpdate}\big(\theta_k, \tilde{\lambda}_k, \widehat{A}_{R,k}, \widehat{A}_{C,k}; \mathcal{B}\big).$$

  $\{$ $\mathcal{B} = \mathrm{FPO}$ for from-scratch training; $\mathcal{B} = \pi_{RL}$ (ODE) for fine-tuning. $\}$
  Define

$$\Delta\theta_k = \theta_k^+ - \theta_k$$

  according to Eq. 13.
  Compute the endpoint cost proxy $\bar{Q}_{C,k}(1)$ and endpoint response S_k using Eq. 15.
  Set $\alpha_k \leftarrow 1$.
  if $k = 0$ **then**
    Initialize the revised multiplier:

$$\tilde{\lambda}_{k+1} \leftarrow \lambda_{k+1}.$$

  else
    Estimate the multiplier–response gain $\widehat{h}_k$ using Eq. 18.
    if $\widehat{h}_k \le -h_{\min}$ **then**
      Response remains informative; update $\tilde{\lambda}_{k+1}$ using Eq. 19.
    else
      Multiplier response is weak; apply a small probe:

$$\tilde{\lambda}_{k+1} = \tilde{\lambda}_k + \delta_\lambda.$$

      if $\bar{Q}_{C,k}(1) > d$ **and** $S_k < 0$ **then**
        Calibrate the cost-decreasing update using Eq. 22.
      end if
    end if
  end if
  Apply the calibrated policy update:

$$\theta_{k+1} = \theta_k + \alpha_k \Delta\theta_k.$$

end for

E.1 Network Architectures

The flow actor consists of a state-context encoder and a velocity network. We append normalized elapsed episode time to the environment observation and denote the resulting $(d_s + 1)$-dimensional input by $\bar{s}$. For flow state x_σ and scalar flow time σ, the actor computes

$$e_s = \tanh(f_{\mathrm{ctx}}(\bar{s})), \tag{33}$$

$$v_\theta(x_\sigma, \sigma; s) = -0.7x_\sigma + f_{\mathrm{vel}}([e_s, x_\sigma, \sigma]). \tag{34}$$

We integrate this velocity field with explicit Euler steps. Reward and cost advantages are obtained from separate state-value networks V_R and V_C. An additional action-conditioned cost critic $Q_C(s, a)$ supplies the endpoint cost gradient used by Flow-Lag. None of these networks share parameters.

All linear layers use orthogonal initialization (Saxe et al., 2013). Hidden-layer weights use gain $\sqrt{2}$ and zero biases. The velocity output layer uses gain 0.01; the value and endpoint-cost output layers

Table 4: Flow-Lag hyperparameters. Values separated by a slash denote the Safe MuJoCo and Safety-Gymnasium settings, respectively.

Hyperparameter	Safe MuJoCo	Safety-Gymnasium
Cost limit d	25	25
Discount factor γ	0.99	0.99
GAE parameter λ_{GAE}	0.95	0.95
Number of parallel environments	16	20
Rollout horizon per environment	1,000	1,000
Actor update epochs	4	3
Critic update epochs	10	10
Minibatch size	1,024	512
Actor learning rate	3×10^{-4}	1×10^{-4}
Critic learning rate	1×10^{-3}	3×10^{-4}
Actor optimizer	Adam (Kingma & Ba, 2014)	Adam
Critic optimizer	Adam	Adam
Gradient-norm limit	1.0	1.0
FPO clipping coefficient	0.2	0.05
Flow integration steps K	8	8
Flow-matching samples per state	8	8
Base distribution p_0	$\mathcal{N}(0, I)$	$\mathcal{N}(0, I)$
Standard dual step size η_λ	0.01	0.01–0.035
Initial standard multiplier λ_0	0	0
Initial revised multiplier $\widetilde{\lambda}_0$	0	0
Target response magnitude ϵ	0.005	0.005
Response fallback step size η_s	0.02	0.02
Minimum reliable gain magnitude $h_{\min}$	0.1	0.1
Sensitivity calibration batch size	2,048	2,048
Step-calibration coefficient β	0.8	0.8
Maximum displacement scale $\alpha_{\max}$	4	4
Observation normalization	No	Yes
Cost-advantage normalization	No	Yes

Table 5: Actor and critic architectures of Flow-Lag. Here d_s and d_a are the observation and action dimensions. "$L \times W$" denotes L hidden layers of width W.

Component	Input	MuJoCo hidden layers	Safety-Gym hidden layers	Activation	Output
Context encoder f_{ctx}	$\bar{s} \in \mathbb{R}^{d_s+1}$	1×64	2×64	Tanh	$e_s \in \mathbb{R}^{64}$
Velocity network f_{vel}	$[e_s, x_\sigma, \sigma] \in \mathbb{R}^{64+d_a+1}$	1×64	2×64	Tanh	$\mathbb{R}^{d_a}$
Reward value V_R	$\bar{s} \in \mathbb{R}^{d_s+1}$	2×64	2×64	Tanh	scalar
Cost value V_C	$\bar{s} \in \mathbb{R}^{d_s+1}$	2×64	2×64	Tanh	scalar
Endpoint cost critic Q_C	$[\bar{s}, a] \in \mathbb{R}^{d_s+d_a+1}$	2×64	2×64	Tanh	scalar

are initialized with zero gain. The flow time is supplied as a raw scalar rather than through a learned or sinusoidal embedding. The reward value, cost value, and endpoint cost critic are optimized jointly but remain parameter-disjoint. When a cost-critic ensemble is enabled, the second Q_C has the same architecture and Flow-Lag uses the elementwise pessimistic estimate $\max(Q_C^{(1)}, Q_C^{(2)})$.