C-STEERING: CONSTRAINED NOISE-SPACE STEERING OF BEHAVIOUR-CLONED FLOW POLICIES FOR OFFLINE SAFE RL

Anonymous authors
Paper under double-blind review

ABSTRACT

Offline safe reinforcement learning must find high-return behaviour within an episodic cost budget from logged data alone. We propose C-Steering, a bounded noise actor over a one-step flow decoder trained only by behaviour cloning. The actor selects actions from the learned decoder's image without a competing cloning penalty. Controlled experiments examine three design choices: cost-budget calibration, decoder-prior geometry and noise-actor parameterisation. Calibrating discounted cost values to the episodic budget improves budget satisfaction substantially more than the tested multiplier-update changes. A bounded full-dimensional prior raises mean return by 12% over a covariance-matched sphere while retaining 37 of 42 budget-satisfying cells, with gains in both return and seed-level safe return supported by seed and task resampling. Additional diagnostics reveal near-boundary noise selection and an approximately 19% larger root-mean-square spread of the decoded boundary image on two-dimensional tasks, characterising how the learned action geometry differs between priors. In a single-task comparison, a direct actor achieves higher return at every seed and lower mean cost than a reflected-flow actor, with two network passes per action instead of eleven. Across 21 DSRL tasks and two budgets, C-Steering meets the budget on 37 cells, 31 with positive return, compared with 27 and 26, respectively, for the baseline with the most budget-satisfying cells. These findings establish an effective constrained-steering design and provide empirical insight into its prior geometry and noise selection.

1 INTRODUCTION

Offline safe reinforcement learning (RL) seeks high return under an expected episodic cost budget, using only logged transitions (Altman, 1999; Liu et al., 2024). A policy must avoid both unsafe behaviour and actions on which its value estimates have little data support. In safety datasets these aims can conflict: across the 21 DSRL datasets considered here, only 16% of logged trajectories meet the tighter evaluated budget, and higher-return trajectories tend to have higher cost (mean correlation 0.52; Appendix F). A behaviour-cloning penalty (Fujimoto & Gu, 2021; Park et al., 2025) can draw a policy toward unsafe average behaviour, while a weak penalty lets it exploit extrapolation error. The relevant opportunity is to select the uncommon safe actions that the dataset does contain.

Latent and noise-space steering offers such a selection mechanism (Zhou et al., 2021; Wagenmaker et al., 2025; Im et al., 2026; Zhang et al., 2026). A generative behaviour model is trained only by behaviour cloning, and a noise actor chooses the input of its decoder; the decoder's weights receive no reward or cost gradients. Every selected action therefore lies in the support of the *modelled* behaviour distribution, so no cloning term is needed in the actor objective and none competes with a cost constraint. Existing steering methods, however, were designed to maximise reward, and this restriction alone does not make the actor safe: an actor trained without the cost constraint over our decoder meets the budget on only 1 of 42 evaluated task–budget cells. Constrained control therefore requires budget calibration and motivates a closer look at how noise-support geometry and noise generation affect safe action selection.

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.

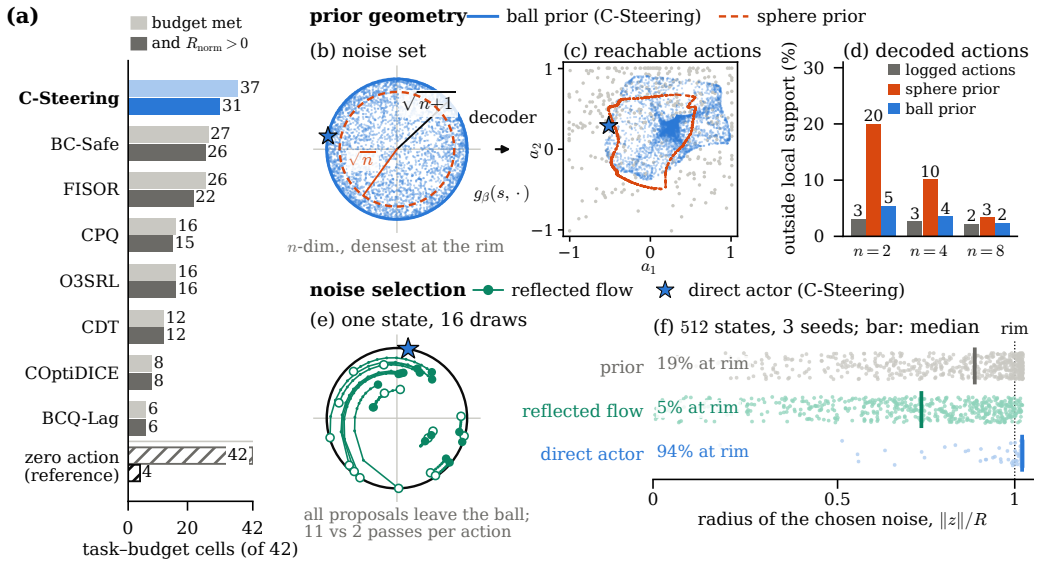


Figure 1: Overview. (a) Cells on which the budget is met, with and without positive normalised return. (b, c) Prior geometry at one CarGoal1 dataset state with two-dimensional actions: spherical noise maps to a curve, while a decoder trained on bounded full-dimensional noise can emit a region of actions; the star is the noise chosen by the actor and its decoded action. (d) Share of decoded actions outside the local support of the logged actions, by action dimension; the diagnostic estimates proximity to logged actions and is not a guarantee of data support. (e, f) Noise selection within the ball by the two actors of Table 3 on PointGoal1 at 500k updates. (e) At one dataset state, ten compensated Euler steps of the reflected flow from 16 prior draws (open circles) to the final noise (filled), and the direct actor's noise (star); every proposed step leaves the ball, so the noise turns about the origin while its radius cannot grow. The state has the median final radius among 512 states. (f) Radius of the prior, of the reflected actor's final noise and of the direct actor's noise over 512 states and three seeds, with medians and the share within 2% of the boundary.

We study three linked design questions. The first is *budget calibration*. The benchmark measures an undiscounted episode sum, whereas temporal-difference (TD) learning estimates a discounted cost value; comparing the two directly permits episode costs larger by $H(1-\gamma)$ under constant per-step costs, a factor of 10 at $\gamma = 0.99$ and $H = 1000$. Threshold conversion (as in OSRL) and per-step budget shifts exist (Liu et al., 2024; Chemingui et al., 2025a), but their effect relative to the multiplier update has not been isolated for this policy class. The second is *the geometry of the noise set*. A sphere in $\mathbb{R}^n$ is $(n-1)$-dimensional, so its image under a Lipschitz decoder has zero n-dimensional volume. This restricts distributional representation, but does not establish a control-performance disadvantage. We test the associated prior choice empirically (Figure 1b–d), with gains concentrated in the two- and four-dimensional tasks of this suite (Figure 13 in Appendix O), and examine the deployed decoder boundaries (Appendix P). The third is *how the noise is generated* within a full-dimensional set. ReFORM (Zhang et al., 2026) selects noise in a ball through ten reflected Euler updates. Its compensated update cannot increase the noise radius (Zhang et al., 2026, Appendix A.3, Eqs. 20–21); repeated activation can therefore restrict radial movement, while each action requires at least eleven sequential network passes. We examine this tradeoff in one constrained task (Figure 1e, f).

We address these questions with **C-Steering**, a noise actor over a one-step behaviour-cloned flow decoder. It trains the cost critic on cost minus the per-step budget, making its sign a budget-relative surrogate rather than a general certificate of episodic safety; it trains the decoder on a bounded prior with an interior, removing the dimensional obstruction without guaranteeing accurate behaviour modelling; and it parameterises the noise directly in that ball with a single actor pass. One configuration is used across 21 DSRL tasks and two budgets. It satisfies the budget on 37 of 42 cells, 31 with positive normalised return (Figure 1a); BC-Safe reaches 27 and 26, respectively, and the principal gains occur on long-horizon Safety-Gymnasium tasks. Controlled ablations locate what

changes the result: calibrated cost targets raise budget satisfaction far more than the tested multiplier updates; the full-dimensional prior improves return over a sphere with the same number, but a different set, of budget-satisfying cells; in a single-task comparison the direct actor achieves higher return and lower cost than a reflected-flow actor while needing two network passes per action instead of eleven; and a behaviour-regularised action-space actor remains competitive in safe return but meets fewer budgets.

Our contributions are (i) a constrained steering method for a one-step behaviour-cloned flow policy, (ii) a controlled study showing the importance of discounted-cost budget calibration in this policy class, and (iii) a theoretical and empirical study of bounded noise priors and actor parameterisations. The latter combines a representation result with robust performance comparisons and diagnostics that reveal near-boundary selection, a larger decoded boundary-image spread, and the behaviour of reflected updates on one task. In an isolated timing comparison, the C-Steering sampling structure takes about one third as long per training update as the ReFORM-style reflected-flow sampling structure (Appendix K).

2 Related Work

Latent and noise-space steering trains a generator on behaviour and optimises its input, while reward gradients do not update the generator's weights (Zhou et al., 2021; Wagenmaker et al., 2025; Im et al., 2026; Zhang et al., 2026). LAPO instead updates an advantage-weighted generative model and a latent policy jointly (Chen et al., 2022). LPS (Im et al., 2026) restricts flow-policy noise to a sphere. ReFORM trains its behaviour flow on a ball, generates the noise with a ten-step reflected flow and distils the behaviour flow into a one-step policy. Both maximise reward. ReFORM reports that a multilayer perceptron (MLP) noise generator and tanh squashing perform worse than its reflected flow on a two-dimensional toy task and the OGBench cube-single environment with noisy data. In our constrained setting, a single-pass actor with a radial tanh map matches a non-saturating radial projection within seed variation (Appendix N) and needs two network passes per action instead of eleven (Appendix K). ReFORM motivates its stochastic generator by multimodal expressivity; our direct actor is deterministic. We compare these actor designs within our constrained backbone, retaining our prior, decoder and critics (Appendix L). This isolates the actor choice in our setting rather than reproducing the complete ReFORM algorithm. LSPC also learns a latent policy for safe offline RL, but its conditional variational autoencoder (CVAE) decoder is cost-weighted rather than trained by behaviour cloning alone (Koirala et al., 2025). OCPO-SA restricts the updates of a parameterised policy to a safe-action distribution that a CVAE distils from the offline data (Hou et al., 2026), and SafeFQL pairs a reachability-based condition with a one-step flow policy (Tayal et al., 2026). Neither imposes an expected-cost budget on an actor over the input of a behaviour model that receives no RL gradients.

Offline safe RL uses Lagrangian value learning, as in BCQ-Lag, which combines offline BCQ (Fujimoto et al., 2019) with a proportional–integral–derivative (PID) Lagrangian (Stooke et al., 2020). Other methods use constraint-penalised Q-learning (Xu et al., 2022), stationary-distribution correction (Lee et al., 2022), feasibility or reachability objectives (Zheng et al., 2024), minimax objectives solved by online optimisation (Chemingui et al., 2025a), safe-trajectory cloning (Liu et al., 2024) and conditioned sequence models (Liu et al., 2023). Threshold conversion for discounted critics (as in OSRL) and a per-step budget shift are established ideas (Liu et al., 2024; Chemingui et al., 2025a), and Sauté RL instead tracks the remaining budget in the state (Sootla et al., 2022). We test threshold conversion and the budget shift, but not state augmentation, against multiplier-rule changes while keeping the constrained steering backbone fixed. This comparison, rather than the cost shift alone, is the contribution of the calibration study.

3 Constrained Noise-Space Steering

3.1 Setting and steering class

Let $\mathcal{D} = \{(s, a, r, c, s')\}$ be a fixed dataset from a behaviour policy π_b with n-dimensional actions $a \in \mathbb{R}^n$, let H be the task's episode length, and let d be the budget on mean undiscounted episodic

cost. The evaluation constraint and the discounted value estimated by TD are distinct:

$$J_C^\pi = \mathbb{E}_\pi\left[\sum_{t=0}^{H-1} c_t\right] \le d, \qquad Q_C^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{T-1} \gamma^t c_t \mid s_0 = s, a_0 = a\right]. \tag{1}$$

Here T ends at a true termination; time-outs do not terminate TD targets. Define Q_R^π analogously for discounted reward. We train a one-step MeanFlow decoder $g_\beta(s,w)$ by behaviour cloning with $w \sim \nu$. The decoder emits one action per query, so the noise w is also n-dimensional. Its behaviour-cloning updates run concurrently with the other networks, but actor and critic gradients never change its weights. An actor $w_\phi(s)$ chooses noise in $\mathcal{B} = \operatorname{supp}\nu$ and emits

$$\pi_\phi(s) = g_\beta(s, w_\phi(s)), \qquad w_\phi(s) \in \mathcal{B}. \tag{2}$$

For continuous g_β and compact $\mathcal{B}$, this action is in $g_\beta(s,\mathcal{B}) = \operatorname{supp}(g_\beta(s,\cdot)_\#\nu)$. This is a statement about the *learned decoder*, not the true data distribution; the proof and local-support diagnostics are in Appendices C and B.

3.2 Calibrating an episodic budget

A direct Lagrangian using Q_C compares quantities with different time weighting. For a policy with constant per-step cost $\bar{c}$, no true termination, and a continuing critic, $Q_C = \bar{c}/(1-\gamma)$ while $J_C = H\bar{c}$. Thus the naive comparison $Q_C \le d$ permits

$$J_C \le H(1-\gamma)d, \tag{3}$$

which can be substantially looser than d. This identity is conditional, not a general relation between a discounted value and an episode sum. OSRL instead rescales the threshold by $\kappa(\gamma,H) = (1-\gamma^H)/(H(1-\gamma))$ (Liu et al., 2024). Under constant per-step cost, this conversion is exact for an H-truncated critic evaluated at the episode start. Applied to our continuing critic under the same condition, it enforces the tighter budget $J_C \le (1-\gamma^H)d$, which at $\gamma = 0.99$ is $0.63d$ for $H{=}100$ and $0.87d$ for $H{=}200$ but above $0.99d$ for $H \ge 500$.

We move the per-step budget into the TD target and learn

$$\widetilde{Q}_C^\pi(s,a) = \mathbb{E}_\pi\left[\sum_{t=0}^{T-1} \gamma^t (c_t - d/H) \mid s_0 = s, a_0 = a\right]. \tag{4}$$

Its TD target is $c - d/H + \gamma m \widetilde{Q}_C(s', \pi_\phi(s'))$, where $m = 0$ only at a true termination. We call this the *amortised cost target*. With constant per-step cost and no true termination, $\widetilde{Q}_C \le 0$ is equivalent to $J_C \le d$ for every H and $\gamma < 1$. With nonuniform or late-occurring costs, discounted and uniform time averages differ, so the sign of $\widetilde{Q}_C$ need not certify the episode budget. The state-dependent offset under true terminations and complete derivations appear in Appendix C. The shift d/H follows the same budget-per-step idea used in O3SRL (Chemingui et al., 2025a); here it trains a separate cost critic for a policy whose decoder receives no RL gradients.

The ideal constrained objective maximises expected episodic reward subject to Eq. (1). In practice, as in offline actor-critic methods, we optimise a dataset-state surrogate:

$$\mathcal{L}_{\text{actor}}(\phi) = -\omega_R \mathbb{E}_{s\sim\mathcal{D}} Q_R(s, \pi_\phi(s)) + \lambda \omega_C \mathbb{E}_{s\sim\mathcal{D}} \widetilde{Q}_C(s, \pi_\phi(s)), \tag{5}$$

$$\lambda \leftarrow \operatorname{clip}\Big(\lambda + \eta_\lambda \mathbb{E}_{s\sim\mathcal{D}} \widetilde{Q}_C(s, \pi_\phi(s)), 0, \lambda_{\max}\Big). \tag{6}$$

The multiplier therefore reacts to values at dataset states, not the distribution of states visited by π_ϕ. Critic error and this distribution shift prevent a general constraint guarantee; Section 5 examines the resulting multiplier saturation.

3.3 A bounded noise set with full dimension

A bounded set prevents the actor from choosing noise norms unseen during decoder training. Its *dimension* also matters. If $\operatorname{supp}\nu$ lies on a compact k-dimensional smooth manifold with $k < n$ and $g_\beta(s,\cdot)$ is Lipschitz, its image has zero n-dimensional Lebesgue measure. If the conditional

behaviour has an absolutely continuous component of mass ρ, the decoder distribution has total-variation distance at least ρ from it (Appendix C). A sphere in $\mathbb{R}^n$ has $k = n-1$; at $n = 2$ its image is a curve even if logged actions locally fill an area.

We project a uniformly sampled sphere of radius $R = \sqrt{n+1}$ in $\mathbb{R}^{n+1}$ onto its first n coordinates. The resulting prior has support on the closed ball $\mathcal{B} = \{w : \|w\| \le R\}$, identity covariance, and density

$$p(w) \propto (R^2 - \|w\|^2)^{-1/2} \quad (\|w\| < R). \tag{7}$$

The actor emits $w_\phi(s) = R\tanh(\|v\|/R)v/\|v\|$, where $v = l_\phi(s)$ and the value at $v = 0$ is defined by continuity. This keeps all chosen noise in $\mathcal{B}$. The construction separates decoder training from noise selection: interior-supported training removes the dimensional obstruction of the spherical prior, while the actor can directly select noise close to the boundary when favoured by the objective. The prior belongs to a projected-sphere family whose limiting boundary case is the sphere and whose members include the uniform ball; sensitivity results are in Appendix B. Our principal comparison matches prior covariance, using sphere radius $\sqrt{n}$ and ball radius $\sqrt{n+1}$. The representation result motivates this design, and Section 4.2 evaluates its control performance and learned boundary geometry.

Support and noise generation. Full-dimensional support does not determine how an actor uses it. For ReFORM's reflected Euler update (Zhang et al., 2026, Eq. 12), let $\hat{z}_{k+1} = z_k + \Delta_k$ and $n_{k+1} = \hat{z}_{k+1}/\|\hat{z}_{k+1}\|$. When the proposal leaves the ball, compensation gives, in exact arithmetic,

$$\begin{aligned} z_{k+1} &= \hat{z}_{k+1} - \langle \Delta_k, n_{k+1}\rangle n_{k+1} \\ &= \langle z_k, n_{k+1}\rangle n_{k+1}, \qquad \|z_{k+1}\| \le \|z_k\|. \end{aligned} \tag{8}$$

Repeated compensation thus prevents radial growth along those steps. Uncompensated steps can increase the radius, so this does not rule out reaching the boundary. Our direct radial map avoids this sequence of updates, although it can still saturate. Appendix L derives the identity and diagnoses its relevance in a controlled experiment.

3.4 Training and evaluated policy

The decoder uses behaviour-cloning loss only. Two reward critics and five cost critics use TD targets; the cost ensemble enters the actor and dual as its mean plus one standard deviation. Critic magnitudes are normalised in the actor loss, while the multiplier uses the raw cost excess. We use projected dual ascent with $\lambda_{\max} = 3$ and evaluate an exponential moving average (EMA) of actor weights. The EMA smooths oscillation near the constraint boundary and does not enter TD targets. All tasks share the same network sizes, optimiser, prior family, multiplier ceiling, and EMA coefficient; only the specified budget d and task horizon H enter the cost target. Algorithm 1 summarises one training update.

Time-outs are not treated as terminals. True terminations are treated as absorbing states, so the TD target stops bootstrapping there and neither the cost nor the shift $-d/H$ accrues afterwards. They occur in four of the 21 datasets, in 29% of the AntCircle episodes and in at most 4% of the episodes of DroneRun, DroneCircle and AntRun. On these tasks the offset in Eq. (9) depends on the state. Under constant per-step cost the amortised test is then conservative rather than exact, and in general it carries no guarantee of sufficiency (Appendix C). The learned policies rarely terminate. Their evaluated episodes last essentially the full 200 or 300 steps on DroneRun, DroneCircle and AntRun, and on average 484 of 500 steps on AntCircle. On AntCircle C-Steering meets the budget in both cells, with the highest return among the methods that do. Full settings and inference timing are in Appendices E and K.

4 Experiments

4.1 Protocol and primary outcomes

We evaluate 21 DSRL tasks (Liu et al., 2024), eight from Bullet-Safety-Gym and thirteen from Safety-Gymnasium, at budgets $d \in \{10, 40\}$. Each C-Steering cell uses five fixed seeds, 500k offline updates and evaluation of the final averaged actor over 20 episodes per seed. The baselines

Algorithm 1 C-Steering training. The budget d and horizon H enter only the cost target; superscript $-$ marks target networks.

Require: dataset $\mathcal{D}$, budget d, horizon H, prior ν on $\mathcal{B}$, ceiling $\lambda_{\max}$, EMA coefficient τ_e

1: initialise decoder β, actor ϕ and its average $\bar{\phi} \leftarrow \phi$, reward critics $Q_R^{1,2}$, cost critics $\widetilde{Q}_C^{1,\dots,5}$, their targets, and $\lambda \leftarrow 1$
2: **for** $k = 1, \dots, K$ **do**
3: sample a minibatch $(s, a, r, c, s', m) \sim \mathcal{D}$ and noise $w \sim \nu$; $m = 0$ only at a true termination
4: $a' \leftarrow g_\beta\big(s', w_\phi(s')\big)$, $w_\phi(s) = R \tanh(\|v\|/R)\, v/\|v\|$ with $v = l_\phi(s)$
5: $y_R \leftarrow r + \gamma m \min_j Q_R^{j-}(s', a')$; $y_C^i \leftarrow c - d/H + \gamma m\, \widetilde{Q}_C^{i-}(s', a')$ for each i Eq. (4)
6: $\widetilde{Q}_C(s, \cdot) \leftarrow \mathrm{mean}_i \widetilde{Q}_C^i(s, \cdot) + \mathrm{std}_i \widetilde{Q}_C^i(s, \cdot)$; $v_C \leftarrow \mathrm{mean}_s \widetilde{Q}_C\big(s, \pi_\phi(s)\big)$
7: one Adam step on: the MeanFlow behaviour-cloning loss of g_β on (s, a, w); the squared TD errors to y_R and y_C^i; and $\mathcal{L}_{\text{actor}}(\phi)$ of Eq. (5) with β and λ held fixed
8: $\lambda \leftarrow \mathrm{clip}(\lambda + \eta_\lambda v_C,\, 0,\, \lambda_{\max})$ Eq. (6)
9: Polyak-update the critic targets; $\bar{\phi} \leftarrow (1 - \tau_e)\bar{\phi} + \tau_e \phi$
10: **end for**
11: **return** $\pi(s) = g_\beta\big(s, w_{\bar{\phi}}(s)\big)$

are BC-Safe (Liu et al., 2024), BCQ-Lag (Fujimoto et al., 2019; Stooke et al., 2020), CPQ (Xu et al., 2022), COptiDICE (Lee et al., 2022), CDT (Liu et al., 2023), FISOR (Zheng et al., 2024) and O3SRL (Chemingui et al., 2025a). We run BC-Safe, BCQ-Lag, CPQ, COptiDICE and CDT from the OSRL implementation (Liu et al., 2024) with its official configurations, and FISOR and O3SRL from their authors' code. All baseline numbers come from our own runs. Against published numbers at the same budget, returns agree and the safety verdict differs on 3 of the 41 comparable cells (Appendix I). The OSRL and O3SRL baselines use their official 100k updates; FISOR uses 1M. The 500k step check covers BC-Safe and BCQ-Lag, not every baseline (Appendix J). Eleven O3SRL tasks required new task entries outside the authors' evaluated set (Appendix I).

A *safe cell* has mean episode cost across seeds no greater than d; we additionally count cells with positive mean normalised return. We report mean return and the mean excess cost $V = \mathrm{mean}_{\text{cells}} \max(0, C/d - 1)$, as well as *seed-level safe return*: $\max(R_{\text{norm}}, 0)$ for a run counts only if that run meets its budget. None of these metrics turns a noisy near-boundary mean into a safety guarantee. Per-task costs, returns and standard deviations are in Table 1 for $d = 10$ and in Table 4 of Appendix A for $d = 40$.

C-Steering meets the budget on 37 cells, versus 27 for BC-Safe and 26 for FISOR (bottom rows of Table 1); it does so with positive return on 31, versus 26 and 22. It is safe on $19/21$ cells at $d = 10$ and $18/21$ at $d = 40$. On Safety-Gymnasium it reaches $23/26$ safe cells, where no baseline exceeds $15/26$; on Bullet its $14/16$ is one fewer than FISOR and O3SRL. The five failed cells and all near-boundary costs are visible in Table 1 and, for $d = 40$, in Table 4. Six safe cells have non-positive return, so satisfaction alone overstates task performance. A zero-action reference policy illustrates the issue: it is safe on every cell but has positive return on only four.

On the 22 cells where C-Steering and FISOR are both safe, C-Steering has higher return on 17 and lower on 4 (one tie); against BC-Safe the split is 18 to 4 (one tie) on 23 jointly safe cells. Its mean normalised return is 0.432 with $V = 0.056$, versus BC-Safe's 0.325 and 0.499 and FISOR's 0.350 and 1.637. At the seed level, safe return is 0.361 with a 95% seed-bootstrap interval of $[0.338, 0.383]$, versus 0.211 for BC-Safe and 0.175 for FISOR; 86% of C-Steering runs meet their budget. These intervals reflect seed variation under this task suite, not uncertainty over new tasks; Appendix O adds intervals that resample tasks.

4.2 Which design choices change the outcome?

Table 2 keeps the training backbone fixed while changing one design choice. Its first three rows use the generous ceiling $\lambda_{\max} = 100$ to isolate cost calibration; later rows use the deployed ceiling 3. The full grid, including all five multiplier rules, multiplier ceilings, noise priors and action-space weights, is in Table 6 of Appendix A.

Table 1: DSRL results at cost limit d=10. Each task spans two rows: normalised return (higher is better) and cost normalised by the budget (lower is better), so cost ≤ 1 *is* constraint satisfaction. Agents that miss the budget are greyed, both rows, since their reward is not comparable; the blue reward is the best among the agents that meet it. Mean $\pm$ std over 5 seeds; safety labels and counts use unrounded means. Averages are given per benchmark suite and over all 21 tasks; an average is greyed when the average cost exceeds the budget. An average return mixes tasks on which a method is safe with tasks on which it is not, so it is reported for comparability with earlier work and not used alone for ranking. The last three rows count, over both budgets (42 cells; Table 4 gives d=40), the cells that meet the budget, those that also have positive mean normalised return, and the Safety-Gymnasium cells (26) that meet the budget.

Task		BC-Safe	BCQ-Lag	CPQ	COptiDICE	CDT	FISOR	O3SRL	C-Steering
Bullet-Safety-Gym									
BallCircle	reward ↑	0.47 ± 0.04	0.72 ± 0.04	0.64 ± 0.02	0.71 ± 0.02	0.70 ± 0.01	0.34 ± 0.01	0.61 ± 0.03	0.76 ± 0.01
	cost ↓	0.70 ± 0.07	3.92 ± 0.32	0.69 ± 0.45	4.48 ± 0.19	1.38 ± 0.08	0.00 ± 0.00	0.15 ± 0.17	0.21 ± 0.20
BallRun	reward ↑	0.27 ± 0.05	0.61 ± 0.21	0.20 ± 0.19	0.57 ± 0.05	0.32 ± 0.01	0.13 ± 0.08	0.23 ± 0.02	0.08 ± 0.09
	cost ↓	3.30 ± 1.48	4.86 ± 2.01	2.25 ± 2.62	5.77 ± 0.43	1.17 ± 0.75	0.74 ± 0.89	0.00 ± 0.00	0.52 ± 1.04
CarCircle	reward ↑	0.34 ± 0.11	0.59 ± 0.05	0.69 ± 0.02	0.46 ± 0.02	0.71 ± 0.01	0.39 ± 0.04	0.67 ± 0.03	0.76 ± 0.01
	cost ↓	1.10 ± 0.87	3.36 ± 0.79	0.00 ± 0.00	4.92 ± 0.64	1.17 ± 0.21	0.16 ± 0.32	0.04 ± 0.05	0.15 ± 0.09
CarRun	reward ↑	0.92 ± 0.07	0.93 ± 0.01	0.89 ± 0.06	0.92 ± 0.02	0.99 ± 0.00	0.69 ± 0.03	0.96 ± 0.01	0.97 ± 0.00
	cost ↓	0.83 ± 1.12	0.19 ± 0.14	1.66 ± 2.31	0.00 ± 0.00	0.77 ± 0.32	0.01 ± 0.02	0.01 ± 0.02	1.02 ± 0.84
DroneCircle	reward ↑	0.49 ± 0.06	0.69 ± 0.05	-0.20 ± 0.02	0.24 ± 0.02	0.57 ± 0.00	0.47 ± 0.04	0.48 ± 0.06	0.58 ± 0.01
	cost ↓	0.39 ± 0.29	4.29 ± 1.43	1.36 ± 0.56	1.51 ± 1.24	1.15 ± 0.14	0.08 ± 0.06	0.14 ± 0.21	0.01 ± 0.02
DroneRun	reward ↑	0.42 ± 0.15	0.71 ± 0.05	0.37 ± 0.29	0.67 ± 0.03	0.58 ± 0.01	0.33 ± 0.04	0.37 ± 0.07	0.57 ± 0.01
	cost ↓	0.86 ± 0.73	8.56 ± 1.64	5.84 ± 4.64	7.37 ± 0.10	0.58 ± 0.35	1.22 ± 0.74	1.08 ± 1.06	0.10 ± 0.19
AntCircle	reward ↑	0.37 ± 0.05	0.56 ± 0.04	0.00 ± 0.00	0.17 ± 0.05	0.46 ± 0.03	0.24 ± 0.05	0.47 ± 0.02	0.56 ± 0.04
	cost ↓	2.35 ± 1.00	4.74 ± 0.71	0.00 ± 0.00	6.09 ± 2.00	3.66 ± 0.82	0.00 ± 0.01	0.00 ± 0.00	0.00 ± 0.00
AntRun	reward ↑	0.57 ± 0.04	0.80 ± 0.03	0.02 ± 0.03	0.61 ± 0.01	0.72 ± 0.01	0.39 ± 0.02	0.29 ± 0.04	0.62 ± 0.06
	cost ↓	1.00 ± 0.31	9.69 ± 1.84	0.02 ± 0.04	2.06 ± 0.83	1.01 ± 0.14	0.15 ± 0.16	0.02 ± 0.02	0.71 ± 0.42
Average (Bullet-Safety-Gym)	reward	0.48	0.70	0.33	0.54	0.63	0.37	0.51	0.61
	cost	1.32	4.95	1.48	4.03	1.36	0.29	0.18	0.34
Budget met (Bullet-Safety-Gym)		4/8	1/8	4/8	1/8	2/8	7/8	7/8	7/8
Safety-Gymnasium									
CarCircle1	reward ↑	0.24 ± 0.04	0.70 ± 0.03	-0.13 ± 0.09	0.71 ± 0.03	0.50 ± 0.03	0.70 ± 0.02	0.57 ± 0.08	0.56 ± 0.03
	cost ↓	3.49 ± 1.57	16.62 ± 0.80	7.44 ± 5.17	20.05 ± 1.00	7.56 ± 1.09	11.06 ± 0.65	7.74 ± 5.47	0.61 ± 0.59
CarCircle2	reward ↑	0.35 ± 0.05	0.65 ± 0.02	0.43 ± 0.11	0.74 ± 0.02	0.63 ± 0.01	0.59 ± 0.04	0.54 ± 0.10	0.49 ± 0.02
	cost ↓	8.73 ± 3.95	18.72 ± 1.88	6.95 ± 6.92	24.77 ± 2.01	13.48 ± 1.87	9.72 ± 3.29	11.59 ± 7.67	0.65 ± 0.69
PointCircle1	reward ↑	0.30 ± 0.09	0.41 ± 0.09	0.43 ± 0.08	0.84 ± 0.02	0.54 ± 0.01	0.48 ± 0.07	0.61 ± 0.05	0.47 ± 0.01
	cost ↓	0.37 ± 0.42	5.31 ± 2.50	0.01 ± 0.02	19.51 ± 0.53	0.66 ± 0.52	15.82 ± 3.32	8.90 ± 1.92	0.00 ± 0.00
PointCircle2	reward ↑	0.36 ± 0.06	0.50 ± 0.04	0.27 ± 0.21	0.86 ± 0.01	0.60 ± 0.01	0.77 ± 0.01	0.47 ± 0.04	0.41 ± 0.03
	cost ↓	2.00 ± 0.92	4.52 ± 1.52	4.02 ± 5.49	29.75 ± 0.95	1.83 ± 0.18	19.21 ± 0.90	2.05 ± 2.01	0.01 ± 0.02
CarGoal1	reward ↑	0.22 ± 0.04	0.42 ± 0.06	0.71 ± 0.06	0.40 ± 0.04	0.64 ± 0.02	0.42 ± 0.05	0.71 ± 0.03	0.00 ± 0.02
	cost ↓	0.92 ± 0.20	2.65 ± 0.57	4.12 ± 1.06	2.19 ± 0.25	3.72 ± 0.46	1.51 ± 0.54	4.47 ± 0.80	0.10 ± 0.07
CarGoal2	reward ↑	0.11 ± 0.02	0.34 ± 0.02	0.48 ± 0.18	0.18 ± 0.03	0.39 ± 0.05	0.02 ± 0.01	0.58 ± 0.12	0.07 ± 0.02
	cost ↓	1.74 ± 0.48	4.70 ± 0.91	15.49 ± 3.32	2.31 ± 0.74	6.18 ± 0.19	0.59 ± 0.64	7.69 ± 2.30	0.54 ± 0.39
PointGoal1	reward ↑	0.25 ± 0.05	0.69 ± 0.02	0.52 ± 0.10	0.45 ± 0.06	0.66 ± 0.04	0.65 ± 0.01	0.76 ± 0.01	0.64 ± 0.03
	cost ↓	0.84 ± 0.26	3.37 ± 0.81	0.60 ± 0.41	6.19 ± 0.89	3.09 ± 0.46	4.55 ± 0.63	3.95 ± 0.80	0.91 ± 0.25
PointGoal2	reward ↑	0.24 ± 0.04	0.68 ± 0.05	0.24 ± 0.15	0.41 ± 0.03	0.43 ± 0.06	0.24 ± 0.05	0.73 ± 0.08	0.11 ± 0.02
	cost ↓	1.62 ± 0.56	10.55 ± 1.27	6.76 ± 2.79	7.00 ± 1.01	5.49 ± 0.93	1.94 ± 0.68	7.90 ± 3.13	2.71 ± 2.74
CarPush2	reward ↑	0.03 ± 0.01	0.14 ± 0.02	0.09 ± 0.13	0.10 ± 0.03	0.17 ± 0.03	0.12 ± 0.04	0.13 ± 0.02	-0.03 ± 0.03
	cost ↓	1.49 ± 0.60	4.58 ± 1.56	14.02 ± 2.65	4.25 ± 0.94	6.60 ± 0.64	1.30 ± 0.70	3.29 ± 0.65	0.09 ± 0.11
PointPush1	reward ↑	0.08 ± 0.01	0.32 ± 0.06	0.17 ± 0.03	0.15 ± 0.02	0.22 ± 0.04	0.30 ± 0.04	0.24 ± 0.06	0.10 ± 0.07
	cost ↓	0.76 ± 0.33	2.73 ± 0.79	1.07 ± 0.44	3.26 ± 0.61	2.91 ± 0.46	1.82 ± 0.16	1.91 ± 0.51	0.14 ± 0.24
CarButton1	reward ↑	0.02 ± 0.01	0.10 ± 0.04	0.34 ± 0.05	-0.04 ± 0.05	0.18 ± 0.06	-0.07 ± 0.06	0.12 ± 0.12	-0.01 ± 0.01
	cost ↓	1.36 ± 0.78	6.55 ± 2.34	39.08 ± 1.48	4.25 ± 1.44	8.22 ± 2.48	0.52 ± 0.32	8.48 ± 6.10	0.50 ± 0.30
CarButton2	reward ↑	-0.02 ± 0.01	0.01 ± 0.05	0.16 ± 0.05	-0.10 ± 0.08	0.19 ± 0.07	-0.00 ± 0.00	0.18 ± 0.22	-0.01 ± 0.01
	cost ↓	2.32 ± 0.57	5.21 ± 1.94	44.02 ± 5.42	3.11 ± 1.41	12.09 ± 2.99	0.32 ± 0.21	11.03 ± 6.30	0.76 ± 0.29
PointButton1	reward ↑	0.02 ± 0.02	0.24 ± 0.03	0.54 ± 0.09	0.11 ± 0.03	0.50 ± 0.04	0.08 ± 0.02	0.40 ± 0.15	-0.16 ± 0.05
	cost ↓	1.08 ± 0.32	5.33 ± 0.84	12.62 ± 1.48	3.46 ± 0.64	10.92 ± 2.14	2.20 ± 0.80	7.97 ± 2.61	0.17 ± 0.09
Average (Safety-Gymnasium)	reward	0.17	0.40	0.33	0.37	0.43	0.33	0.46	0.20
	cost	2.06	6.99	12.02	10.01	6.37	5.43	6.69	0.55
Budget met (Safety-Gymnasium)		4/13	0/13	2/13	0/13	1/13	3/13	0/13	12/13
Average	reward ↑	0.29	0.51	0.33	0.44	0.51	0.35	0.48	0.36
	cost ↓	1.77	6.21	8.00	7.73	4.46	3.47	4.21	0.47
Budget met		8/21	1/21	6/21	1/21	3/21	10/21	7/21	19/21
Budget met, both budgets		27/42	6/42	16/42	8/42	12/42	26/42	16/42	**37/42**
and $R_{\text{norm}} > 0$		26/42	6/42	15/42	8/42	12/42	22/42	16/42	**31/42**
Budget met, Safety-Gymnasium, both budgets		15/26	3/26	5/26	4/26	4/26	11/26	1/26	**23/26**

Calibration dominates the tested multiplier-rule changes. With uncalibrated $Q_C \leq d$ and $\lambda_{\max} = 100$, plain dual ascent meets the budget on $7/42$ cells. Five update rules on the same target span 7–15 safe cells; either a rescaled threshold or the amortised target with plain dual ascent reaches 38 or 37. The rescaled threshold differs from the uncalibrated rows only in the threshold of the dual update. Its critic target and the normaliser of its cost term are the same, so the change from 7 to 38 safe cells isolates the threshold. The median episode cost of the uncalibrated policy increases with H, while remaining below the loosened bound of Eq. (3) (Figure 2 in Appendix B). These results identify calibration as the larger effect *within the tested backbone and rules*; they do not imply that a multiplier rule cannot matter elsewhere. The amortised target changes the threshold, the critic target and the normaliser of the cost term together (Remark 3 in Appendix C), so its

Table 2: Key ablations over all 42 cells, five seeds each. V is mean excess cost. The first three rows use $\lambda_{\max} = 100$ and the others use 3. See Table 6 for all settings and safe-return values.

Configuration	Safe	Safe, $R > 0$	Mean R	$V \downarrow$
Uncalibrated Q_C, threshold d	7	7	0.654	3.005
Rescaled threshold κd	38	29	0.319	0.036
Amortised target, $\lambda_{\max} = 100$	37	28	0.317	0.011
C-Steering, $\lambda_{\max} = 3$	**37**	31	0.432	0.056
Sphere prior, same actor averaging	37	32	0.387	0.114
Noise actor not bounded	26	20	0.327	0.542
Action actor, cloning weight $\alpha = 0.1$	31	29	0.456	0.279
No cost constraint	1	1	0.805	8.482

similarity to the rescaled threshold combines the threshold with a change of optimisation scale. The shifted TD target incorporates a termination-dependent offset, but its sign does not generally certify the episodic budget.

The full-dimensional prior improves return and safe return. Holding critic architecture, constraint settings and actor averaging fixed, the full-dimensional prior configuration raises mean return from 0.387 to 0.432, an approximately 12% improvement over the sphere. The paired difference is $+0.045$, with seed-bootstrap interval $[+0.033, +0.057]$ and task-bootstrap interval $[+0.017, +0.075]$. Seed-level safe return also increases by $+0.036$ (approximately 11%), with seed and task intervals $[+0.007, +0.066]$ and $[+0.002, +0.074]$. Both configurations meet the budget on $37/42$ cells, although the safe sets differ. The full-dimensional prior achieves higher return on $28/42$ cells, and its gains are concentrated in two- and four-dimensional tasks (Figure 13 in Appendix O). Its observed excess cost is also lower (0.056 versus 0.114), but this difference is sensitive to a few tasks and its interval includes zero; the robust advantages are return and safe return (Appendix O). Removing the noise-actor bound reduces safe cells from 37 to 26, further supporting bounded steering. Sensitivity across bounded priors is reported in Appendix B.

Interior-supported training coexists with near-boundary selection. The actor chooses noise within 2% of the boundary on approximately 95% of dataset states. This pattern also appears with a non-saturating radial projection actor (97%), suggesting that it is not solely an artefact of the tanh parameterisation (Appendix N). In the three-seed diagnostic, projecting ball-actor noise onto the exact boundary changes its own critic's feasibility label on only 0.2% of sampled states in the two-dimensional group, with the same feasible fraction at the reported precision. Thus the observed benefit of the ball configuration coexists with predominantly near-boundary selection, motivating examination of the learned boundary map as well as the interior. Action-change tails and the scope of this statewise diagnostic are reported in Appendix P.

The ball decoder has a broader boundary image. Across the 34 two-dimensional cells, its decoded boundary image has root-mean-square (RMS) spread 0.56 versus 0.47 for the sphere, an approximately 19% increase (difference $+0.09$, task-bootstrap interval $[+0.04, +0.14]$). This shows that near-boundary selection need not give the same action geometry under the two priors: the ball-trained decoder's boundary spans a broader range in this RMS sense. Selected ball actions are also less often flagged outside nearby logged actions (14.0% versus 17.6%), while lying farther from the local action mean (1.02 versus 0.88 in units of local spread). Together, these measurements characterise a broader boundary image and a distinct pattern of action selection. The difference is not greater angular uniformity: unevenness is slightly higher, arc length has no clear difference, and predicted-feasibility comparisons depend on the scoring critic. The larger image spread is therefore a concrete geometric observation accompanying the return gain, with its causal role left open (Appendix P).

Direct versus reflected noise generation. On PointGoal1 at $d = 10$, we replace our actor by a ReFORM-style ten-step reflected flow while retaining the same full-dimensional prior, MeanFlow decoder and constrained-learning settings. Both variants use gradient-norm clipping at 10; the reflected update uses a numerically stable form of Eq. (8). Over three seeds, the direct actor has higher

Table 3: Noise-actor comparison on PointGoal1, $d = 10$: means over three seeds, final 500k update, 20 evaluation episodes per seed. Both actors use gradient clipping. Safe return counts an over-budget seed as zero.

Noise actor	$C/d \downarrow$	$R_{\text{norm}} \uparrow$	Safe return $\uparrow$
Direct (C-Steering + clipping)	0.958	0.661	0.428
Reflected flow	1.208	0.362	0.124

final return and lower mean cost (Table 3). It is also cheaper to run: it needs two network passes per action instead of eleven, and a complete training run took 27 rather than 47 minutes on average (Appendix L). At all six diagnostic checkpoints, every tested Euler proposal of the evaluated EMA reflected actor leaves the ball, activating compensation. At 500k, only 4.8% of the reflected actor's sampled noise lies within 2% of the boundary, versus 94.3% for the direct actor. These observations are consistent with inhibited radial growth, but do not establish that it causes the performance gap. Appendix L reports the protocol, per-seed results and diagnostics; this is an actor ablation on one task, not a full ReFORM benchmark.

Alternative actors and operating points. An action-space actor with the same critics and cost target and a behaviour-cloning weight $\alpha = 0.1$ reaches 31 safe cells, six fewer than C-Steering, and has five times its mean excess cost. Its seed-level safe return is nevertheless comparable (0.366 versus 0.361), so C-Steering's advantage is greater safety coverage, not dominance on every return metric. The multiplier ceiling sets an operating point: $\lambda_{\max} = 10$ meets the budget on 40 cells, versus 37 at 3, but has a lower observed seed-level safe return (0.332 versus 0.361). We use 3 for the latter criterion; this ceiling, the prior and actor averaging were selected using this evaluation suite, as the action-space actor's cloning weight would have to be. Appendix H reports a post-hoc within-suite analysis of the ceiling choice, not an independent test of transfer to new task families.

5 Limitations and Conclusion

The bounded noise actor restricts actions to the learned decoder's image, not to the unknown support of the true behaviour policy. The local-support diagnostic flags a nonzero share of selected actions outside nearby logged actions (Appendix B). The cost constraint is a discounted critic surrogate averaged over dataset states. In saturated runs, the critic's estimated violation is more common at dataset states than at states visited by the learned policy, so increasing the multiplier can reduce return despite safe evaluated episodes. Late-concentrated costs, critic error, and noisy near-boundary evaluations remain further sources of safety error. The present benchmark evidence is therefore an empirical improvement, not a safety certificate.

The geometric diagnostics characterise the learned policies but do not isolate the cause of the prior's return gain. The covariance-matched priors differ in maximal radius and radial training distribution, and the three-seed boundary measurements use final checkpoints at dataset states. Same-radius retraining and projected-policy rollouts remain useful interventions; the full scope is discussed in Appendix P.

C-Steering combines calibrated cost learning, a bounded full-dimensional decoder prior and efficient direct noise selection. Across 42 task–budget cells it meets 37 budgets, more than any evaluated external baseline. Controlled comparisons show that calibration is the dominant tested factor in budget satisfaction, while the full-dimensional prior improves mean return and seed-level safe return over the sphere. The additional diagnostics enrich this picture: interior-supported decoder training coexists with near-boundary actor selection, and the resulting boundary image has approximately 19% greater RMS spread on two-dimensional tasks. Finally, the direct actor outperforms the reflected-flow actor in the controlled single-task comparison with fewer network passes. These results provide both performance evidence and a more detailed empirical account of constrained noise-space steering.

Reproducibility statement

Appendix E and Table 7 give the single configuration used for all 42 task–budget cells, including network sizes, optimiser settings, the noise prior, the multiplier update and the evaluation protocol, and Appendix F lists the episode lengths that enter the cost target. Every C-Steering cell uses seeds 100–104 and reports the final checkpoint without selection. Appendix E also gives the implementation, configuration and seeds of each baseline, and Appendix I compares our baseline runs with published numbers. The proofs are in Appendix C, the statistical procedures in Appendix O and the protocol of the reflected-flow comparison in Appendix L. The anonymized supplementary material contains the training code of C-Steering and the scripts that launch the experiments.

AI use statement

In this work, we used generative AI tools to help develop the conceptual framing of the study, to formulate mathematical claims and assist in writing the proofs in the appendix, to propose and refine hypotheses for the diagnostic analyses, to provide feedback on the design of experiments and ablations, to implement methods and analysis code (including the reflected-flow ablation and the diagnostic scripts), to interpret results, and to translate the authors' notes into English. We have not used generative AI tools to generate synthetic data sets or to clean or reformat datasets, and qualitative or thematic data analysis is not applicable to this work. Additionally, we used generative AI tools to create or edit software code and scientific figures, to draft parts of the paper and edit it for readability, to suggest its structure, to search for, identify and summarise related literature, and to format references. We have reviewed all AI-assisted work: every reported number was traced to the raw result files and recomputed with independent scripts, and every reference was checked against its published version. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

References

Eitan Altman. *Constrained Markov Decision Processes*. Chapman and Hall/CRC, 1999.

Yassine Chemingui, Aryan Deshwal, Alan Fern, Thanh Nguyen-Tang, and Jana Doppa. Online optimization for offline safe reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 38, pp. 140670–140698, 2025a.

Yassine Chemingui, Aryan Deshwal, Honghao Wei, Alan Fern, and Jana Doppa. Constraint-adaptive policy switching for offline safe reinforcement learning. In *AAAI Conference on Artificial Intelligence*, volume 39, pp. 15722–15730, 2025b.

Xi Chen, Ali Ghadirzadeh, Tianhe Yu, Jianhao Wang, Alex Yuan Gao, Wenzhe Li, Bin Liang, Chelsea Finn, and Chongjie Zhang. LAPO: Latent-variable advantage-weighted policy optimization for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 36902–36913, 2022.

Scott Fujimoto and Shixiang Shane Gu. A minimalist approach to offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 20132–20145, 2021.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2052–2062. PMLR, 2019.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Advances in Neural Information Processing Systems*, volume 38, pp. 75460–75482, 2025.

Diyuan Hou, Longyang Huang, Pu Feng, and Wenjun Wu. Offline constrained policy optimization with safe anchoring. *Neural Networks*, 201:108865, 2026.

Hokyun Im, Andrey Kolobov, Jianlong Fu, and Youngwoon Lee. Latent policy steering through one-step flow policies. In *Proceedings of Robotics: Science and Systems*, Sydney, Australia, July 2026. doi: 10.15607/RSS.2026.XXII.152.

Prajwal Koirala, Zhanhong Jiang, Soumik Sarkar, and Cody Fleming. Latent safety-constrained policy approach for safe offline reinforcement learning. In *International Conference on Learning Representations*, 2025. arXiv:2412.08794.

Jongmin Lee, Cosmin Paduraru, Daniel J. Mankowitz, Nicolas Heess, Doina Precup, Kee-Eung Kim, and Arthur Guez. COptiDICE: Offline constrained reinforcement learning via stationary distribution correction estimation. In *International Conference on Learning Representations*, 2022.

Zuxin Liu, Zijian Guo, Yihang Yao, Zhepeng Cen, Wenhao Yu, Tingnan Zhang, and Ding Zhao. Constrained decision transformer for offline safe reinforcement learning. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 21611–21630. PMLR, 2023.

Zuxin Liu, Zijian Guo, Haohong Lin, Yihang Yao, Jiacheng Zhu, Zhepeng Cen, Hanjiang Hu, Wenhao Yu, Tingnan Zhang, Jie Tan, and Ding Zhao. Datasets and benchmarks for offline safe reinforcement learning. *Journal of Data-centric Machine Learning Research*, 1(12):1–29, 2024.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. In *International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48104–48127. PMLR, 2025.

Aivar Sootla, Alexander I. Cowen-Rivers, Taher Jafferjee, Ziyan Wang, David H. Mguni, Jun Wang, and Haitham Bou-Ammar. Sauté RL: Almost surely safe reinforcement learning using state augmentation. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 20423–20443. PMLR, 2022.

Adam Stooke, Joshua Achiam, and Pieter Abbeel. Responsive safety in reinforcement learning by PID Lagrangian methods. In *International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9133–9143. PMLR, 2020.

Mumuksh Tayal, Manan Tayal, and Ravi Prakash. Safe flow Q-learning: Offline safe reinforcement learning with reachability-based flow policies. *Reinforcement Learning Journal*, 7, 2026. URL https://rlj.cs.umass.edu/2026/papers/Paper82.html.

Andrew Wagenmaker, Yunchu Zhang, Mitsuhiko Nakamoto, Seohong Park, Waleed Yagoub, Anusha Nagabandi, Abhishek Gupta, and Sergey Levine. Steering your diffusion policy with latent space reinforcement learning. In *Proceedings of the 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 258–282. PMLR, 2025.

Haoran Xu, Xianyuan Zhan, and Xiangyu Zhu. Constraints penalized Q-learning for safe offline reinforcement learning. In *AAAI Conference on Artificial Intelligence*, volume 36, pp. 8753–8760, 2022.

Songyuan Zhang, Oswin So, H. M. Sabbir Ahmad, Eric Yang Yu, Matthew Cleaveland, Mitchell Black, and Chuchu Fan. ReFORM: Reflected flows for on-support offline RL via noise manipulation. In *International Conference on Learning Representations*, 2026.

Yinan Zheng, Jianxiong Li, Dongjie Yu, Yujie Yang, Shengbo Eben Li, Xianyuan Zhan, and Jingjing Liu. Safe offline reinforcement learning with feasibility-guided diffusion model. In *International Conference on Learning Representations*, 2024.

Wenxuan Zhou, Sujay Bajracharya, and David Held. PLAS: Latent action space for offline reinforcement learning. In *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pp. 1719–1735. PMLR, 2021.

A Detailed results and diagnostic ablations

This appendix reports the per-task results at $d = 40$ (those at $d = 10$ are in Table 1) and the full ablation grid behind the compact summaries in Tables 1 and 2.

Table 4: DSRL results at cost limit d=40. Each task spans two rows: normalised return (higher is better) and cost normalised by the budget (lower is better), so cost ≤ 1 *is* constraint satisfaction. Agents that miss the budget are greyed, both rows, since their reward is not comparable; the blue reward is the best among the agents that meet it. Mean $\pm$ std over 5 seeds; safety labels and counts use unrounded means. Averages are given per benchmark suite and over all 21 tasks; an average is greyed when the average cost exceeds the budget. An average return mixes tasks on which a method is safe with tasks on which it is not, so it is reported for comparability with earlier work and not used alone for ranking (§4).

Task		BC-Safe	BCQ-Lag	CPQ	COptiDICE	CDT	FISOR	O3SRL	C-Steering
Bullet-Safety-Gym									
BallCircle	reward ↑	0.58 ± 0.04	0.71 ± 0.05	0.68 ± 0.04	0.70 ± 0.02	0.84 ± 0.00	0.34 ± 0.01	0.72 ± 0.06	0.89 ± 0.02
	cost ↓	0.52 ± 0.12	0.99 ± 0.14	0.82 ± 0.19	1.13 ± 0.04	0.96 ± 0.02	0.00 ± 0.00	0.47 ± 0.31	0.92 ± 0.08
BallRun	reward ↑	0.35 ± 0.11	0.60 ± 0.28	0.30 ± 0.02	0.55 ± 0.04	0.53 ± 0.01	0.13 ± 0.08	0.25 ± 0.01	0.04 ± 0.09
	cost ↓	0.78 ± 0.36	1.17 ± 0.68	0.72 ± 0.24	1.42 ± 0.07	0.99 ± 0.02	0.15 ± 0.20	0.00 ± 0.00	0.00 ± 0.00
CarCircle	reward ↑	0.55 ± 0.08	0.75 ± 0.03	0.73 ± 0.02	0.46 ± 0.03	0.79 ± 0.01	0.41 ± 0.03	0.66 ± 0.04	0.78 ± 0.03
	cost ↓	0.79 ± 0.20	1.24 ± 0.34	0.34 ± 0.16	1.25 ± 0.31	0.84 ± 0.06	0.02 ± 0.03	0.01 ± 0.02	0.65 ± 0.47
CarRun	reward ↑	0.97 ± 0.00	0.94 ± 0.03	0.96 ± 0.01	0.93 ± 0.01	0.98 ± 0.00	0.70 ± 0.03	0.96 ± 0.00	0.98 ± 0.01
	cost ↓	0.17 ± 0.19	0.02 ± 0.02	0.89 ± 0.51	0.00 ± 0.00	0.54 ± 0.09	0.02 ± 0.01	0.00 ± 0.00	0.52 ± 0.32
DroneCircle	reward ↑	0.59 ± 0.02	0.93 ± 0.01	-0.22 ± 0.02	0.27 ± 0.03	0.68 ± 0.01	0.45 ± 0.03	0.57 ± 0.03	0.64 ± 0.07
	cost ↓	0.58 ± 0.10	2.33 ± 0.04	0.41 ± 0.31	0.39 ± 0.22	0.98 ± 0.03	0.01 ± 0.01	0.35 ± 0.24	0.65 ± 0.62
DroneRun	reward ↑	0.34 ± 0.19	0.68 ± 0.04	0.45 ± 0.17	0.65 ± 0.04	0.66 ± 0.03	0.34 ± 0.04	0.39 ± 0.05	0.66 ± 0.07
	cost ↓	0.68 ± 0.44	2.49 ± 0.49	2.54 ± 0.97	1.81 ± 0.04	1.00 ± 0.11	0.45 ± 0.17	0.54 ± 0.68	1.31 ± 0.61
AntCircle	reward ↑	0.42 ± 0.03	0.71 ± 0.05	0.01 ± 0.01	0.18 ± 0.04	0.52 ± 0.06	0.21 ± 0.05	0.48 ± 0.05	0.81 ± 0.04
	cost ↓	0.62 ± 0.28	2.03 ± 0.45	0.00 ± 0.00	1.43 ± 0.61	1.28 ± 0.21	0.00 ± 0.01	0.00 ± 0.01	0.53 ± 0.18
AntRun	reward ↑	0.68 ± 0.04	0.75 ± 0.09	0.02 ± 0.02	0.62 ± 0.02	0.73 ± 0.01	0.43 ± 0.03	0.23 ± 0.14	0.69 ± 0.02
	cost ↓	0.85 ± 0.10	2.02 ± 0.59	0.00 ± 0.00	0.46 ± 0.15	0.86 ± 0.10	0.06 ± 0.05	0.02 ± 0.02	0.50 ± 0.18
Average (Bullet-Safety-Gym)	reward	0.56	0.76	0.37	0.55	0.72	0.38	0.53	0.69
	cost	0.62	1.54	0.72	0.99	0.93	0.09	0.17	0.64
Budget met (Bullet-Safety-Gym)		8/8	2/8	7/8	3/8	6/8	8/8	8/8	7/8
Safety-Gymnasium									
CarCircle1	reward ↑	0.35 ± 0.05	0.74 ± 0.01	0.16 ± 0.26	0.69 ± 0.01	0.57 ± 0.02	0.68 ± 0.02	0.52 ± 0.07	0.55 ± 0.03
	cost ↓	1.00 ± 0.28	4.61 ± 0.35	2.36 ± 2.18	4.92 ± 0.35	2.59 ± 0.32	2.86 ± 0.34	1.14 ± 0.52	0.14 ± 0.15
CarCircle2	reward ↑	0.56 ± 0.03	0.73 ± 0.03	0.44 ± 0.12	0.75 ± 0.02	0.63 ± 0.03	0.61 ± 0.05	0.49 ± 0.05	0.52 ± 0.02
	cost ↓	2.96 ± 0.50	5.99 ± 0.62	3.72 ± 2.45	6.35 ± 0.50	3.50 ± 0.49	2.84 ± 1.29	1.62 ± 0.63	0.25 ± 0.20
PointCircle1	reward ↑	0.39 ± 0.09	0.48 ± 0.17	0.43 ± 0.07	0.83 ± 0.04	0.57 ± 0.01	0.51 ± 0.06	0.61 ± 0.05	0.48 ± 0.02
	cost ↓	0.07 ± 0.07	2.02 ± 1.54	0.58 ± 1.15	4.88 ± 0.16	0.81 ± 0.16	3.47 ± 1.21	2.25 ± 0.52	0.07 ± 0.09
PointCircle2	reward ↑	0.48 ± 0.03	0.69 ± 0.03	0.26 ± 0.21	0.87 ± 0.01	0.62 ± 0.01	0.74 ± 0.04	0.51 ± 0.05	0.43 ± 0.04
	cost ↓	1.42 ± 0.39	3.95 ± 0.60	1.72 ± 1.19	7.64 ± 0.28	1.22 ± 0.11	4.18 ± 0.32	1.47 ± 0.74	0.00 ± 0.00
CarGoal1	reward ↑	0.23 ± 0.02	0.44 ± 0.04	0.69 ± 0.12	0.40 ± 0.04	0.65 ± 0.03	0.43 ± 0.04	0.71 ± 0.02	0.75 ± 0.02
	cost ↓	0.22 ± 0.09	0.64 ± 0.12	1.25 ± 0.19	0.52 ± 0.07	1.21 ± 0.14	0.42 ± 0.13	1.28 ± 0.20	1.02 ± 0.18
CarGoal2	reward ↑	0.14 ± 0.04	0.31 ± 0.05	0.66 ± 0.09	0.25 ± 0.03	0.47 ± 0.05	0.04 ± 0.02	0.70 ± 0.04	0.30 ± 0.03
	cost ↓	0.49 ± 0.15	1.47 ± 0.36	3.67 ± 0.65	0.87 ± 0.26	1.83 ± 0.22	0.15 ± 0.10	2.26 ± 0.29	0.70 ± 0.15
PointGoal1	reward ↑	0.37 ± 0.06	0.72 ± 0.02	0.55 ± 0.09	0.49 ± 0.04	0.69 ± 0.03	0.66 ± 0.02	0.76 ± 0.01	0.78 ± 0.01
	cost ↓	0.38 ± 0.09	0.99 ± 0.10	0.27 ± 0.13	1.55 ± 0.11	0.90 ± 0.16	1.05 ± 0.17	1.12 ± 0.11	0.91 ± 0.08
PointGoal2	reward ↑	0.32 ± 0.02	0.71 ± 0.02	0.35 ± 0.06	0.38 ± 0.03	0.51 ± 0.05	0.28 ± 0.07	0.79 ± 0.04	0.65 ± 0.09
	cost ↓	0.87 ± 0.24	2.43 ± 0.21	1.36 ± 0.45	1.51 ± 0.30	1.69 ± 0.25	0.63 ± 0.19	2.78 ± 0.60	0.94 ± 0.17
CarPush2	reward ↑	0.04 ± 0.03	0.14 ± 0.04	0.23 ± 0.08	0.08 ± 0.04	0.22 ± 0.03	0.13 ± 0.03	0.16 ± 0.04	0.28 ± 0.03
	cost ↓	0.35 ± 0.12	1.08 ± 0.26	4.18 ± 0.55	1.03 ± 0.35	1.55 ± 0.36	0.19 ± 0.12	1.35 ± 0.58	1.27 ± 0.63
PointPush1	reward ↑	0.15 ± 0.05	0.27 ± 0.04	0.24 ± 0.08	0.13 ± 0.02	0.28 ± 0.06	0.30 ± 0.04	0.31 ± 0.02	0.39 ± 0.09
	cost ↓	0.56 ± 0.14	0.66 ± 0.05	0.48 ± 0.26	0.68 ± 0.19	0.65 ± 0.12	0.61 ± 0.26	0.72 ± 0.06	0.39 ± 0.19
CarButton1	reward ↑	0.03 ± 0.01	0.08 ± 0.05	0.51 ± 0.04	-0.09 ± 0.07	0.18 ± 0.02	-0.04 ± 0.05	0.26 ± 0.14	-0.01 ± 0.01
	cost ↓	0.47 ± 0.08	1.63 ± 0.42	7.34 ± 0.77	1.36 ± 0.28	2.45 ± 0.65	0.14 ± 0.09	3.01 ± 2.03	0.10 ± 0.07
CarButton2	reward ↑	-0.01 ± 0.05	0.01 ± 0.03	0.35 ± 0.07	-0.12 ± 0.08	0.18 ± 0.08	-0.01 ± 0.02	0.17 ± 0.18	-0.01 ± 0.01
	cost ↓	0.79 ± 0.37	1.50 ± 0.50	9.76 ± 1.45	1.08 ± 0.49	3.15 ± 0.65	0.13 ± 0.05	2.44 ± 1.53	0.04 ± 0.03
PointButton1	reward ↑	0.05 ± 0.02	0.20 ± 0.04	0.72 ± 0.05	0.10 ± 0.03	0.56 ± 0.02	0.08 ± 0.01	0.56 ± 0.02	0.03 ± 0.03
	cost ↓	0.44 ± 0.15	1.17 ± 0.24	2.91 ± 0.19	0.88 ± 0.30	2.34 ± 0.13	0.47 ± 0.15	2.73 ± 0.36	0.18 ± 0.03
Average (Safety-Gymnasium)	reward	0.24	0.42	0.43	0.37	0.47	0.34	0.50	0.40
	cost	0.77	2.16	3.05	2.56	1.84	1.32	1.86	0.46
Budget met (Safety-Gymnasium)		11/13	3/13	3/13	4/13	3/13	8/13	1/13	11/13
Average	reward ↑	0.36	0.55	0.41	0.43	0.56	0.35	0.51	0.50
	cost ↓	0.72	1.92	2.16	1.96	1.49	0.85	1.22	0.53
Budget met		19/21	5/21	10/21	7/21	9/21	16/21	9/21	18/21

Table 5: Constraint satisfaction rate over all evaluated cells, by budget and by task family, and mean normalised return on the cells where a method is safe. The third column counts cells that meet the budget *and* have positive normalised return; the two reference policies at the bottom show why it is needed. $\bar{R}_{\text{norm}}|\text{safe}$ is a conditional mean over each method's *own* safe cells and is therefore not comparable across rows. Computed from Tables 1–4. SG denotes Safety-Gymnasium and Btn Button.

Method	Budget met	met & $R_{\text{norm}}>0$	d=10	d=40	Bullet	SG-Circle	SG-Goal	SG-Push/Btn	$\bar{R}_{\text{norm}}\|$safe
BC-Safe	64.3% (27/42)	26/42	38.1%	90.5%	75.0%	37.5%	75.0%	60.0%	+0.360
BCQ-Lag	14.3% (6/42)	6/42	4.8%	23.8%	18.8%	0.0%	25.0%	10.0%	+0.672
CPQ	38.1% (16/42)	15/42	28.6%	47.6%	68.8%	25.0%	25.0%	10.0%	+0.376
COptiDICE	19.0% (8/42)	8/42	4.8%	33.3%	25.0%	0.0%	25.0%	20.0%	+0.452
CDT	28.6% (12/42)	12/42	14.3%	42.9%	50.0%	25.0%	12.5%	10.0%	+0.682
FISOR	61.9% (26/42)	22/42	47.6%	76.2%	93.8%	0.0%	50.0%	70.0%	+0.262
O3SRL	38.1% (16/42)	16/42	33.3%	42.9%	93.8%	0.0%	0.0%	10.0%	+0.517
C-Steering	**88.1%** (37/42)	**31/42**	90.5%	85.7%	87.5%	100.0%	75.0%	90.0%	+0.416
reference policies (not methods)									
zero action	100.0% (42/42)	4/42	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	−0.119
decoder only, $w \sim \nu$	11.9% (5/42)	5/42	0.0%	23.8%	12.5%	0.0%	25.0%	10.0%	+0.483

Table 6: Ablations. Every row uses one configuration for all 42 cells (5 seeds, five cost critics, full-dimensional prior, averaged actor unless stated); rows differ only in the component named. "Met & $R>0$" counts cells that meet the budget with positive normalised return; $\bar{R}$ is the mean normalised return and V the mean excess cost $\max(0, C/d - 1)$ over all 42 cells; "Safe ret." is the seed-level safe return, in which a run that violates its budget counts as zero. The last column counts, over cells where the row and the reference are both safe, the cells on which the row has the higher / lower return; the reference is the amortised target with dual ascent at $\lambda_{\max}$=100 for blocks (a)–(d) and C-Steering for blocks (e)–(f).

Cost target	Component	Budget met	Met & $R>0$	d=10	d=40	$\bar{R}$	$V\downarrow$	Safe ret.	W/L vs ref.
(a) threshold calibration; dual ascent, $\lambda_{\max}$=100									
standard, τ=d	dual ascent	16.7% (7/42)	7	28.6%	4.8%	+0.654	3.005	0.101	6/1
standard, τ=$\kappa(\gamma, H)d$	dual ascent	90.5% (38/42)	29	90.5%	90.5%	+0.319	0.036	0.304	16/20
amortised, τ=0	dual ascent (reference)	88.1% (37/42)	28	95.2%	81.0%	+0.317	0.011	0.305	—
(b) multiplier rule with the uncalibrated threshold; $\lambda_{\max}$=100									
standard, τ=d	optimistic	16.7% (7/42)	7	23.8%	9.5%	+0.653	3.173	0.104	6/1
standard, τ=d	PID	26.2% (11/42)	11	38.1%	14.3%	+0.648	3.043	0.126	9/2
standard, τ=d	augmented	35.7% (15/42)	15	47.6%	23.8%	+0.589	2.465	0.102	4/11
standard, τ=d	scale-aware	21.4% (9/42)	9	33.3%	9.5%	+0.646	3.114	0.137	6/3
(c) multiplier rule with the amortised target; $\lambda_{\max}$=100									
amortised, τ=0	optimistic	90.5% (38/42)	30	85.7%	95.2%	+0.324	0.044	0.291	17/18
(d) multiplier ceiling; amortised target, dual ascent									
amortised, τ=0	$\lambda_{\max}$=30	90.5% (38/42)	29	90.5%	90.5%	+0.339	0.038	0.310	23/13
amortised, τ=0	$\lambda_{\max}$=10	95.2% (40/42)	32	100.0%	90.5%	+0.383	0.012	0.332	26/11
amortised, τ=0	$\lambda_{\max}$=3 (**C-Steering**)	**88.1%** (37/42)	**31**	90.5%	85.7%	+0.432	0.056	0.361	26/10
amortised, τ=0	$\lambda_{\max}$=1	69.0% (29/42)	28	57.1%	81.0%	+0.517	1.314	0.345	16/11
(e) noise set, evaluated actor and constraint; otherwise as C-Steering									
amortised, τ=0	sphere prior, averaged actor	88.1% (37/42)	32	85.7%	90.5%	+0.387	0.114	0.324	11/23
amortised, τ=0	full-dimensional prior, last iterate	88.1% (37/42)	31	85.7%	90.5%	+0.426	0.035	0.342	15/19
amortised, τ=0	sphere prior, last iterate	81.0% (34/42)	31	76.2%	85.7%	+0.384	0.181	0.277	10/21
amortised, τ=0	noise actor not bounded	61.9% (26/42)	20	71.4%	52.4%	+0.327	0.542	0.281	6/19
none	no constraint ($\lambda \equiv 0$)	2.4% (1/42)	1	0.0%	4.8%	+0.805	8.482	0.012	—
(f) policy class: behaviour-regularised action-space actor with weight α, averaged; critics, target and dual as C-Steering									
amortised, τ=0	α=0	64.3% (27/42)	23	66.7%	61.9%	+0.381	0.622	0.254	11/13
amortised, τ=0	α=0.01	54.8% (23/42)	21	71.4%	38.1%	+0.469	0.351	0.263	17/4
amortised, τ=0	α=0.1	73.8% (31/42)	29	61.9%	85.7%	+0.456	0.279	0.366	16/12
amortised, τ=0	α=1	66.7% (28/42)	27	52.4%	81.0%	+0.426	0.628	0.309	9/15
amortised, τ=0	α=10 (last iterate)	35.7% (15/42)	15	19.0%	52.4%	+0.445	2.453	0.166	5/7

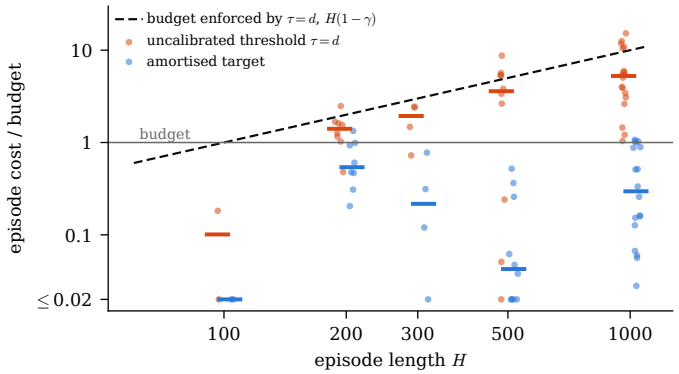


Figure 2: Prediction and observation for the threshold. Each point is one task–budget cell (mean of 5 seeds) trained with dual ascent at $\lambda_{\max}$=100; bars are medians over the cells with the same episode length. With the uncalibrated threshold the median episodic cost increases with H while remaining below the loosened bound $H(1-\gamma)\,d$ of Proposition 3 (dashed); with the amortised target the median stays below the budget at every H.

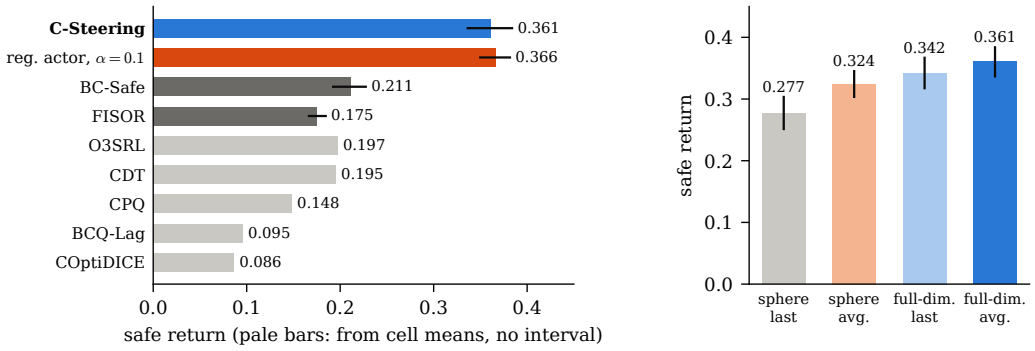


Figure 3: Seed-level safe return: the normalised return of a run counts only if the run meets its budget. *Left:* methods; black segments are 95% bootstrap intervals over seeds where per-seed results are available. *Right:* the noise set and the evaluated actor, varied independently.

B Additional diagnostic figures

Remark 1 (Projected-sphere prior family). Projecting a uniform sphere with m extra coordinates gives $\|w\|^2/R^2 \sim \mathrm{Beta}(n/2, m/2)$. As $m \to 0$ the law approaches a sphere in $\mathbb{R}^n$, the prior of LPS (Im et al., 2026). $m = 1$ gives Eq. (7), and $m = 2$ is uniform in the ball, the prior of ReFORM (Zhang et al., 2026). With $R = \sqrt{n+m}$, the limit $m \to \infty$ is a standard Gaussian. The ablations keep $R = \sqrt{n+1}$ fixed while varying m. The evaluated bounded members and radius variations are compared in Figure 7.

Local-support diagnostic. For a query (s, a), the diagnostic takes the $K = 100$ dataset states nearest to s in standardised observation space, from a fixed pool of 150,000 dataset states, together with their logged actions. The action a is counted as outside the local support when its distance to the nearest of these logged actions exceeds the 95th percentile of the leave-one-out nearest-neighbour distances among the logged actions themselves; the typicality score is the distance from a to their mean in units of their root-mean-square spread. The rate for logged actions (3% in Figure 1d) is therefore a calibration reference set by the threshold, not an independent measurement of data quality. The diagnostic estimates proximity to logged actions at nearby states. It does not establish the support of the conditional behaviour distribution at s, and proximity to logged actions is not a guarantee of safety.

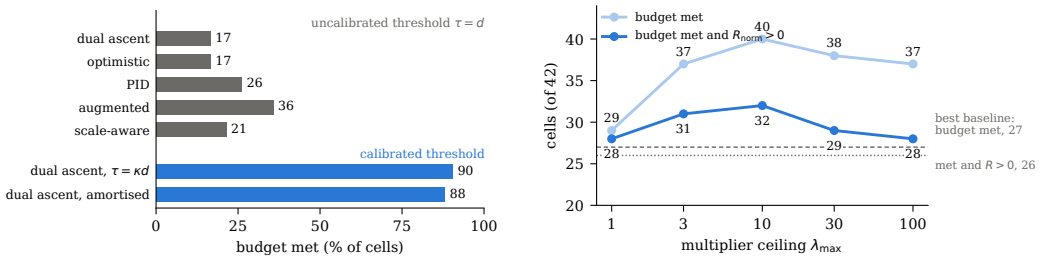


Figure 4: *Left:* share of cells on which the budget is met, for five multiplier update rules with the uncalibrated threshold and for dual ascent with either calibration (Table 6, blocks (a) and (b)). *Right:* effect of the multiplier ceiling on the number of cells that meet the budget, with and without the positive-return condition; dashed lines are the best baseline for each count (block (d)).

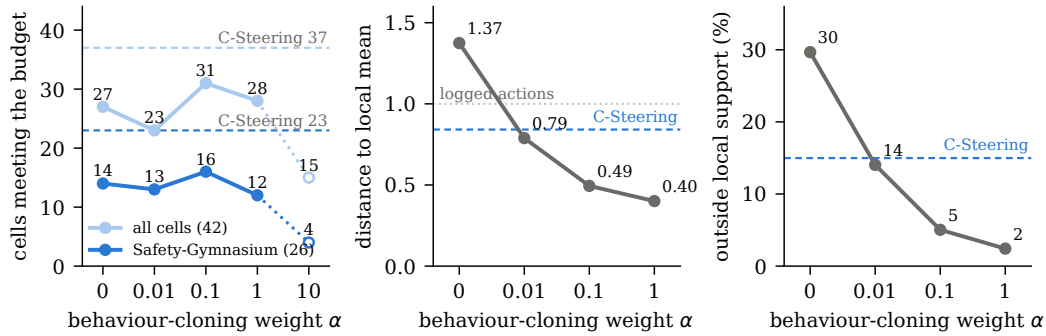


Figure 5: A behaviour-regularised action-space actor trained with the critics, cost target and dual update of C-Steering, as a function of its penalty weight α (Table 6, block (f)). *Left:* cells that meet the budget; dashed lines are C-Steering, and the open marker at α=10 is evaluated without weight averaging. *Centre, right:* distance of the policy's actions from the local mean of the logged actions, in units of their spread, and share of its actions outside the local support of the data, over rollouts at d=10 on all 21 tasks.

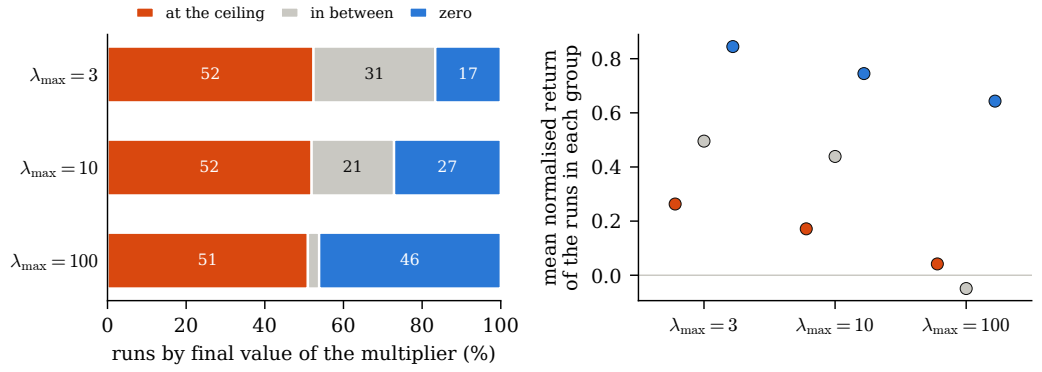


Figure 6: The multiplier at the end of training, for three ceilings. *Left:* share of the 210 runs whose multiplier ends at the ceiling, at zero, or in between. *Right:* mean normalised return of the runs in each group.

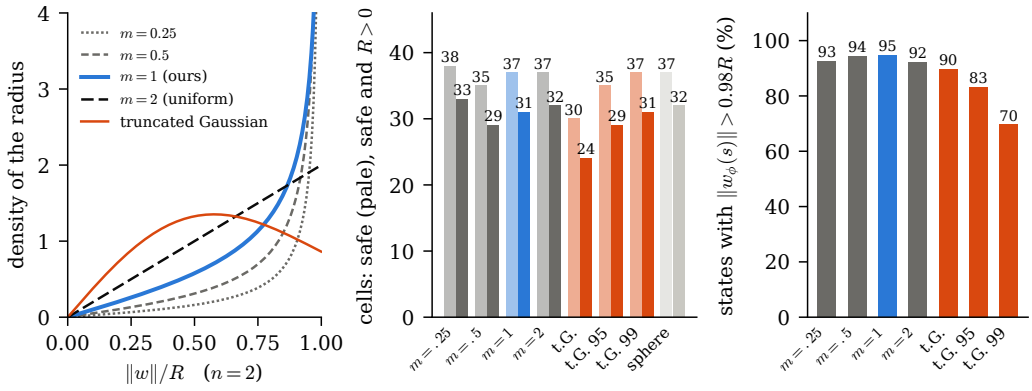


Figure 7: The prior family of Remark 1. *Left:* density of the radius $\|w\|/R$ for n=2. *Centre:* safe cells and safe cells with positive return for each prior (t.G.: Gaussian truncated at $\sqrt{n+1}$, or at its 95% / 99% quantile). *Right:* share of dataset states at which the trained actor places its noise within 2% of the boundary.

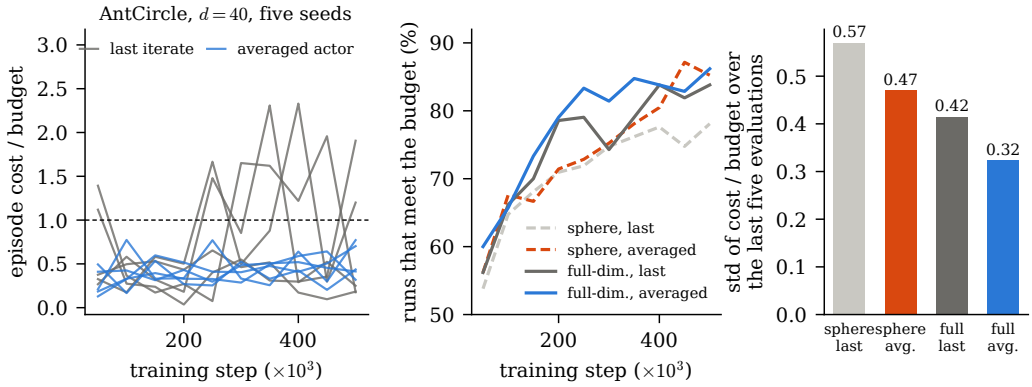


Figure 8: Last iterate against averaged actor. *Left:* evaluated cost over training on one cell, five seeds each. *Centre:* share of the 210 runs that meet the budget at each evaluation. *Right:* standard deviation of the normalised cost over a run's last five evaluations, averaged over runs.

C PROOFS

For reference, we state the conditional results used in the main text before their proofs.

Proposition 1 (Support containment). *If $g_\beta(s,\cdot)$ is continuous and $\mathcal{B} = \operatorname{supp}\nu$ is compact, then $g_\beta(s,\mathcal{B}) = \operatorname{supp}(g_\beta(s,\cdot)_\#\nu)$ and $\pi_\phi(s) \in g_\beta(s,\mathcal{B})$ for every steering actor $w_\phi(s) \in \mathcal{B}$.*

Proposition 2 (Dimension restriction). *Let $\operatorname{supp}\nu$ lie in a compact smooth manifold of dimension $k < n$, and let $g_\beta(s,\cdot)$ be Lipschitz. Then $g_\beta(s,\operatorname{supp}\nu)$ has zero n-dimensional Lebesgue measure. If $\pi_b(\cdot \mid s)$ has an absolutely continuous component of mass ρ, then $\mathrm{TV}(g_\beta(s,\cdot)_\#\nu, \pi_b(\cdot \mid s)) \geq \rho$.*

Proposition 3 (Threshold mismatch under uniform cost). *For a continuing critic with constant per-step cost $\bar{c}$, no true termination, and a task episode of length H, $Q_C = \bar{c}/(1-\gamma)$ and $J_C = H\bar{c}$. Consequently $Q_C \leq d$ is equivalent to Eq. (3).*

Definition 1 (Amortised cost value). *The value $\widetilde{Q}^\pi_C$ is defined by the shifted-cost target in Eq. (4).*

Proposition 4 (Scope of the amortised threshold). *For T steps until true termination,*

$$\widetilde{Q}^\pi_C(s,a) = Q^\pi_C(s,a) - \frac{d}{H}\,\mathbb{E}_\pi\left[\sum_{t=0}^{T-1}\gamma^t \mid s,a\right]. \tag{9}$$

Without true termination and under constant per-step cost $\bar{c}$, $\widetilde{Q}^\pi_C \leq 0$ if and only if $J^\pi_C \leq d$. With an earlier true termination and the same cost assumption, the test is sufficient but conservative. With nonuniform cost, it is in general neither necessary nor sufficient for the episodic budget.

Proposition 1. $\mathcal{B} = \operatorname{supp}\nu$ is compact and $g_\beta(s,\cdot)$ is continuous, so $g_\beta(s,\mathcal{B})$ is compact, hence closed, and carries all the mass of $\mu_\beta(\cdot \mid s) = g_\beta(s,\cdot)_\#\nu$; conversely, if an open set U contains a point $g_\beta(s,w)$ with $w \in \mathcal{B}$, then $g_\beta(s,\cdot)^{-1}(U)$ is an open neighbourhood of $w \in \operatorname{supp}\nu$, so $\mu_\beta(U \mid s) = \nu(g_\beta(s,\cdot)^{-1}(U)) > 0$. Therefore $\operatorname{supp}\mu_\beta(\cdot \mid s) = g_\beta(s,\mathcal{B})$, and $w_\phi(s) \in \mathcal{B}$ by construction. □

Proposition 2. A compact smooth k-manifold has finite k-dimensional Hausdorff measure, and a Lipschitz map with constant L increases $\mathcal{H}^k$ by at most the factor L^k; hence $\mathcal{H}^k\big(g_\beta(s,\operatorname{supp}\nu)\big) < \infty$, and a set of finite $\mathcal{H}^k$ measure with $k < n$ is Lebesgue-null in $\mathbb{R}^n$. Let $E = g_\beta(s,\operatorname{supp}\nu)$. Then $\mu_\beta(E \mid s) = 1$, while the absolutely continuous component of $\pi_b(\cdot \mid s)$ gives E no mass, so $\pi_b(E \mid s) \leq 1-\rho$ and $\mathrm{TV} \geq \mu_\beta(E \mid s) - \pi_b(E \mid s) \geq \rho$. □

Proposition 3. With time-outs not treated as terminals the TD fixed point is the continuing-task value, and with per-step cost constant at $\bar{c}$, $Q^\pi_C = \bar{c}\sum_{t\geq 0}\gamma^t = \bar{c}/(1-\gamma)$, while the episodic cost is $J^\pi_C = \bar{c}H$. Hence $Q^\pi_C = J^\pi_C/(H(1-\gamma))$, and $Q^\pi_C \leq d$ is equivalent to Eq. (3). The factor $H(1-\gamma)$ exceeds one if and only if $H > 1/(1-\gamma)$. □

For comparison, a critic truncated at H (time-outs treated as terminals) evaluated at the first state of an episode gives $Q^\pi_C = \bar{c}\,(1-\gamma^H)/(1-\gamma) = \kappa(\gamma,H)\,J^\pi_C$, which is the conversion used by OSRL. Applied to our continuing critic, $\tau = \kappa(\gamma,H)\,d$ enforces $J^\pi_C \leq (1-\gamma^H)\,d$, which is tighter than the budget where H is small (Remark 2).

Proposition 4. The TD target that defines $\widetilde{Q}_C$ is $(c - d/H) + \gamma\, m\, \widetilde{Q}_C(s',a')$ with bootstrap mask $m = \mathbb{1}[s' \text{ is a true terminal}]^c$. Time-outs are not terminals, so $m = 0$ only at a true termination (§3.4). Unrolling the target, both the cost and the shift are summed up to the termination time T: $\widetilde{Q}^\pi_C = \mathbb{E}_\pi[\sum_{t<T}\gamma^t(c_t - d/H)]$ and $Q^\pi_C = \mathbb{E}_\pi[\sum_{t<T}\gamma^t c_t]$, which gives Eq. (9); with $T = \infty$ the offset is $(d/H)/(1-\gamma)$. Under a constant per-step cost $\bar{c}$ up to termination, $\widetilde{Q}^\pi_C = (\bar{c} - d/H)\,\mathbb{E}_\pi[\sum_{t<T}\gamma^t]$, whose sign is that of $\bar{c} - d/H$ for any $T \geq 1$. Without termination $J^\pi_C = \bar{c}H$, so $\widetilde{Q}^\pi_C \leq 0$ if and only if $J^\pi_C \leq d$; with termination at $T < H$, $J^\pi_C = \bar{c}T$, so $\widetilde{Q}^\pi_C \leq 0$ implies $J^\pi_C \leq (T/H)\,d$. Without termination and in general, multiplying Eq. (9) by $(1-\gamma)$ gives $(1-\gamma)\widetilde{Q}^\pi_C = (1-\gamma)\,\mathbb{E}_\pi[\sum_{t\geq 0}\gamma^t c_t] - d/H$, while $J^\pi_C \leq d$ reads $\frac{1}{H}\,\mathbb{E}_\pi[\sum_{t<H} c_t] - d/H \leq 0$; the two left-hand sides differ by the difference of the two averages stated in the proposition. □

Remark 2 (Amortisation versus threshold rescaling). Keeping Q_C and setting $\tau = \kappa(\gamma,H)\,d$ subtracts $(d/H)(1-\gamma^H)/(1-\gamma)$ from the quantity compared against zero, whereas amortisation subtracts $(d/H)/(1-\gamma)$; the two differ by the factor $(1-\gamma^H)^{-1}$, which at $\gamma = 0.99$ is 1.58 for H=100,

1.16 for H=200, 1.01 for H=500 and 1.00004 for H=1000. On the long-horizon tasks the two thresholds are numerically indistinguishable and can separate only where H is small; Remark 3 lists the other differences between the two implementations. Had time-outs been treated as terminals, the amortised target would instead subtract the state-dependent offset $(d/H)(1-\gamma^h)/(1-\gamma)$ at a state with h steps to go, and the two would separate at every H; the distinction is an implementation choice, not a property of the objective.

Remark 3 (When the two calibrations coincide). Without true termination, Eq. (9) reads $\widetilde{Q_C} = Q_C - b$ with $b = d/(H(1-\gamma))$. Suppose both critics are exact and the standard critic is compared with $\tau = b$. The dual signal $\mathbb{E}_{s\sim\mathcal{D}} Q_C(s, \pi_\phi(s)) - b$ then equals $\mathbb{E}_{s\sim\mathcal{D}} \widetilde{Q_C}(s, \pi_\phi(s))$, and the action gradients $\nabla_a Q_C$ and $\nabla_a \widetilde{Q_C}$ are equal. The unnormalised Lagrangian therefore gives the same actor update for the same multiplier. Four things separate the two implementations in practice. First, the rescaled threshold of OSRL is $\kappa(\gamma, H)\, d = (1-\gamma^H)\, b$ rather than b (Remark 2). Second, a true termination makes the offset in Eq. (9) depend on the state and action (Proposition 4). Third, both critics are learned by TD with function approximation, and nothing forces the learned values to stay exactly b apart during training. Fourth, the actor loss of Eq. (5) divides the cost term by the batch mean of the absolute value of the critic in use. This mean is $\mathrm{mean}|Q_C|$ for the standard target and $\mathrm{mean}|Q_C - b|$ for the amortised target. For the same multiplier, the two targets therefore weight the cost gradient differently relative to the reward gradient, by the factor $\mathrm{mean}|Q_C|/\mathrm{mean}|Q_C - b|$. This factor is large near the constraint boundary, where $Q_C \approx b$ on most states, and below one where Q_C is much smaller than b. The multiplier uses the unnormalised violation and is not affected by this normaliser. In Table 6(a), the uncalibrated and rescaled-threshold rows share the standard target and its normaliser and differ only in τ. The amortised row differs from them in the threshold, the critic target and the normaliser.

D Design space of alternatives

The main text fixes the amortised cost target (Definition 1) and plain projected dual ascent. This appendix catalogues the alternatives we also trained and evaluated; their results appear in Table 6. §3.2 motivates comparing the variants in §D.1 with those in §D.2, and §4.2 reports the comparison.

D.1 Multiplier update rules

Equation 6 is one choice among several. We study:

- **Optimistic.** For convex–concave saddle problems, dual ascent converges only in the averaged iterate while deployment uses the last one, and optimistic mirror descent restores last-iterate convergence; our actor–critic problem is not of that form, so the rule is a heuristic here. It extrapolates, $\lambda \leftarrow \mathrm{clip}(\lambda + \eta_\lambda(2v_t - v_{t-1}), 0, \lambda_{\max})$.
- **PID** (Stooke et al., 2020). Treats v as a control error, $\lambda_t = \mathrm{clip}(K_P v_t + K_I I_t + K_D(v_t - v_{t-1}), 0, \lambda_{\max})$ with integral $I_t = \max(0, I_{t-1} + v_t)$, damping the overshoot of pure integral control. The original rule uses a one-sided derivative $\max(0, v_t - v_{t-1})$ and no cap $\lambda_{\max}$.
- **Augmented Lagrangian.** Adds an adaptive quadratic penalty $\frac{\rho_t}{2}\max(0, v)^2$, giving gradient signal before λ has risen.
- **Scale-aware.** Rescales the step with the budget, $\eta_\lambda \to \eta_\lambda d^p$, since cost magnitudes grow with d.

D.2 Cost-critic targets

Orthogonally, we vary what Q_C regresses.

- **Standard.** $Q_C \approx \sum_t \gamma^t c_t$, constraint $Q_C \le d$. This compares a *discounted* value against an *undiscounted* episodic budget, so the threshold has to be converted by a factor depending on γ and H (Proposition 3).
- **Amortised.** $Q_C \approx \sum_t \gamma^t (c_t - d/H)$ with constraint $Q_C \le 0$: the critic regresses the per-step budget *excess*, absorbing d into the target and turning the decision into a sign test.

This mirrors O3SRL's $r' = r - \lambda(c - d/H)$ but applied to the cost critic of a noise-space actor.

- **Rescaled threshold.** Standard target with $\tau = \kappa(\gamma, H)\, d$ (Remark 2); evaluated in §4.2.

E Implementation details

C-Steering is run with the configuration of Table 7 on every task and budget. The only quantities that change between the 42 cells are the budget d and the episode length H (Appendix F), which enter the cost target $c_t - d/H$ and nothing else. The implementation builds on the public LPS code base (Im et al., 2026) in JAX; the ablations of §4.2 change one entry of the table at a time.

Networks. The decoder g_β is a one-step MeanFlow model (Geng et al., 2025): an MLP with four hidden layers of 512 units, without layer normalisation, that maps the state, a noise vector and the MeanFlow time pair to an action. It is trained on the MeanFlow behaviour-cloning loss; time pairs (r, t) are drawn from a logit-normal distribution (location -0.4, scale 1), and 75% of each batch uses $r = t$ (plain flow matching) and 25% uses $r < t$, with noise $w \sim \nu$ from the prior of Eq. (7) (radius $R = \sqrt{n+1}$), and it receives no gradient from the critics or the actor. The noise actor is an MLP with two hidden layers of 256 units whose output $v \in \mathbb{R}^n$ enters the ball through the radial map $w = R \tanh(\|v\|/R)\, v/\|v\|$. The critics are MLPs with four hidden layers of 256 units and layer normalisation. There are two reward critics, whose TD target takes the minimum of the two target networks and whose mean enters the actor loss, and five cost critics, trained on the shifted cost $c_t - d/H$ with independent TD targets and aggregated as mean plus one standard deviation both in the actor loss and in the dual signal.

Optimisation. All networks use Adam with learning rate 3×10^{-4}, no weight decay and no gradient clipping, batches of 256 transitions, and one update per batch for 500k updates; target networks follow the online networks with Polyak coefficient 0.005; both critics use $\gamma = 0.99$. The scale normalisers of Eq. (5) are $\omega_R = 1/\mathrm{mean}_{\mathrm{batch}}|Q_R|$ and $\omega_C = 1/\mathrm{mean}_{\mathrm{batch}}|\widetilde{Q_C}|$, treated as constants, so the actor loss is $-\omega_R \,\mathrm{mean}\, Q_R + \lambda\, \omega_C \,\mathrm{mean}\, \widetilde{Q_C}$. The multiplier starts at $\lambda = 1$ and is updated at every step by projected dual ascent, Eq. (6), with step $\eta_\lambda = 0.01$ on the raw (unnormalised) violation $v = \mathrm{mean}_{\mathrm{batch}} \widetilde{Q_C}$ and ceiling $\lambda_{\max} = 3$. Time-outs are not treated as terminals. The exponential moving average of the actor's weights used at evaluation has coefficient $\tau_e = 5 \times 10^{-4}$ and does not enter training.

Evaluation. Every 50k updates the averaged actor is rolled out for 20 episodes; the paper reports the evaluation at the final update (500k), with no checkpoint selection. Returns are normalised with the DSRL reference values of each task and costs are divided by the budget. The seeds are 100–104 for every cell.

Compute. One run takes about 35 minutes of wall-clock time with four runs sharing one NVIDIA GH200, which is about 0.15 GPU-hours per run. The 210 runs of the main configuration take about 30 GPU-hours and the 22 ablation rows of Table 6, 4620 runs, about 700 GPU-hours. Software: Python 3.10, JAX 0.4.35, Flax 0.8.5, Optax 0.2.3; gymnasium 0.28.1, safety-gymnasium 0.4.0, bullet-safety-gym 1.4.0, MuJoCo 2.3.0. The datasets are the official DSRL releases, used without filtering or augmentation.

Baselines. BC-Safe, BCQ-Lag, CPQ, COptiDICE and CDT are run from the OSRL implementation with its official configurations for 100k updates (batch 512; 2048 for CDT), seeds 0–4; O3SRL from its authors' code for 100k updates with TD3+BC as the inner solver and its default bandit settings (five arms, multiplier learning rate 0.002, update period 10), seeds 10–14; FISOR from its authors' code for 1M updates with its paper hyperparameters (16 candidates, 5 denoising steps, expectile 0.9), seeds 100–104. Each is evaluated at its final checkpoint over 20 episodes. CDT's training does not depend on the budget, so one model per task and seed is conditioned at evaluation on target cost d and on a target return from its official configuration, which lists several (return, cost) pairs per task. We use the return of the pair whose cost is nearest to d. FISOR's training also does not depend on the budget, and its results at the two budgets are two independently trained sets of the same model (Appendix I). Appendix I lists what their configurations set per task.

Table 7: The single configuration of C-Steering, used for all 42 task–budget cells.

Component	Setting
Decoder g_β	one-step MeanFlow, MLP 4×512, no layer norm, $r < t$ on 25% of pairs
Noise prior ν	Eq. (7): projected sphere, $R = \sqrt{n+1}$ ($m = 1$ in Remark 1)
Noise actor w_ϕ	MLP 2×256, radial map $R \tanh(\|v\|/R)\, v/\|v\|$
Reward critics	$2\times$ MLP 4×256 with layer norm; min for targets, mean in the actor loss
Cost critics	$5\times$ MLP 4×256 with layer norm; independent targets; mean $+\,1{\cdot}$std
Cost target	amortised, $c_t - d/H$, threshold $\tau = 0$; time-outs not terminal
Discount, Polyak	$\gamma = 0.99$ (both critics), target-network coefficient 0.005
Optimiser	Adam, learning rate 3×10^{-4}, no weight decay, no gradient clipping
Batch, updates	256; 500k updates, one per batch
Multiplier	dual ascent, $\lambda_0 = 1$, $\eta_\lambda = 0.01$, $\lambda_{\max} = 3$, updated every step
Critic scale normalisers	$\omega_R = 1/\text{mean}\|Q_R\|$, $\omega_C = 1/\text{mean}\|\widetilde{Q_C}\|$ (stop-gradient)
Evaluated actor	EMA of weights, $\tau_e = 5 \times 10^{-4}$; deterministic action
Evaluation	every 50k updates, 20 episodes; final update reported; seeds 100–104
Per-task quantities	budget $d \in \{10, 40\}$ and episode length H only

F Episode lengths

Episode lengths are those of the DSRL/OSRL task definitions. $\kappa(\gamma, H) = (1-\gamma^H)/(H(1-\gamma))$ at $\gamma = 0.99$ is the rescaling used by OSRL; $H(1-\gamma)$ is the factor by which $\tau = d$ loosens the intended budget for our continuing critic (Proposition 3), and $(1-\gamma^H)^{-1}$ is the factor separating the amortised offset from the rescaled threshold (Remark 2).

Tasks	H	$H(1-\gamma)$	$\kappa(\gamma, H)$	$(1-\gamma^H)^{-1}$
BallRun	100	1	0.634	1.578
BallCircle, CarRun, DroneRun, AntRun	200	2	0.433	1.155
CarCircle, DroneCircle	300	3	0.317	1.052
AntCircle, CarCircle1/2, PointCircle1/2	500	5	0.199	1.007
CarGoal1/2, PointGoal1/2, CarPush2, PointPush1, CarButton1/2, PointButton1	1000	10	0.100	1.00004

Dataset statistics. The two statistics in the introduction are computed from the logged trajectories of the 21 DSRL datasets. For each dataset we take the share of trajectories whose episodic cost is at most 10, the tighter evaluated budget, and the Pearson correlation between the episodic return and the episodic cost of its trajectories. Averaged over the datasets, these are 16% and 0.52.

G Further discussion

The multiplier saturates on some tasks. With the ceiling at $\lambda_{\max}$=100 the multiplier reaches it and stays there in about half of the runs, on 20 of the 42 cells in at least four of five seeds. Those runs are safe, but their return is that of the zero-action policy. The cause is the state average noted in §3.2. In the saturated cells, 45% of dataset states have $\widetilde{Q_C}(s, \pi_\phi(s)) > 0$ under the learned policy, against 9% of the states that policy itself visits; the dual update sees the former, so its violation estimate never becomes non-positive although the policy's episodic cost is far below the budget. The same cost critic separates violating from non-violating runs when it is averaged over the policy's own states (area under the receiver operating characteristic curve, AUC 0.70–0.78 across the 210 runs) and does not when averaged over dataset states (AUC 0.41). Lowering the ceiling, as we do, compensates for this without removing it. Three changes to the dual's estimate that we tried at $\lambda_{\max}$=100 did not remove it either: evaluating it on trajectory-start states (24 of 42 cells safe), using the ensemble mean (33), and restricting it to dataset states whose preceding behaviour agrees with the policy (19); each moved the operating point rather than the trade-off. These statistics were measured with the sphere prior and last-iterate evaluation; the noise prior does not enter the dual update. Estimating the violation on-policy from offline data is open.

Where other actors are as good. With the calibrated target, an action-space actor at its best penalty weight matches C-Steering in seed-level safe return (0.366 against 0.361) and has the higher return on 16 of the 28 cells where both are safe (Table 6(f)). What it gives up is six safe cells and a five times larger excess cost, and its weight has to be selected, which cannot be done offline. On Bullet, C-Steering is safe on 14 of 16 cells, one fewer than FISOR and O3SRL; its advantage is on the Safety-Gymnasium tasks with $H \geq 500$. Six of its safe cells have non-positive return.

The action set is fixed by the decoder. The class Π_β cannot leave the image of g_β. The decoder alone is safe on 5 of 42 cells, so steering does find the safe part of that image, but a base policy that rarely produces safe behaviour bounds what steering can reach.

Protocol. The ceiling $\lambda_{\max}$=3 is one value for all cells but was chosen on the evaluation metric, as were the noise prior and the averaging of the actor; Table 6(d)–(e) and the sensitivity analysis of §4.2 give the other settings. Appendix H repeats the choice of the ceiling with four tasks held out at a time. Baselines use their official configurations without further tuning on our side, and their official step counts (100k against our 500k; training BC-Safe and BCQ-Lag for 500k steps does not change the comparison, Appendix J), and 11 of O3SRL's 21 tasks lie outside the set its authors evaluated (§4). The offset identity of Eq. (9) holds for any cost sequence, with or without true terminations. The threshold relation of Proposition 3 and the equivalence between the amortised sign test and the episodic budget in Proposition 4 require a constant per-step cost and no true termination. With terminations and a constant per-step cost the sign test is conservative; with non-uniform cost it carries no general guarantee of sufficiency (§3.4).

H SELECTING THE MULTIPLIER CEILING WITH TASKS HELD OUT

The ceiling $\lambda_{\max} = 3$ was chosen after all 21 tasks had been run, by the seed-level safe return used throughout the paper. To test how much this choice depends on the tasks it was made on, we repeat it with tasks held out. The analysis is post hoc and uses the runs of Table 6(a) and (d) without new training. For each of the $\binom{21}{4} = 5{,}985$ ways to remove four tasks, both budgets of a removed task are held out together. The ceiling is selected from $\{1, 3, 10, 30, 100\}$ by mean safe return on the remaining 17 tasks and evaluated on the eight held-out cells. The criterion was fixed before this analysis, but it was itself chosen with all 21 tasks in view.

Table 8: Selection of the multiplier ceiling by seed-level safe return with tasks held out. Both budgets of a held-out task are held out together. "Budget met" and "safe return" are measured on the held-out cells. The first row averages over all 5,985 four-task splits; each of the next four rows holds out one group of tasks. The last row is the selection on all tasks, as reported in the main text.

Held-out tasks	Selected $\lambda_{\max}$	Budget met	Safe return
Four tasks, all 5,985 splits	3 (97.6%), 1 (2.4%)	7.0/8	0.358
Bullet-Safety-Gym (8 tasks)	3	14/16	0.550
Safety-Gymnasium Circle (4 tasks)	3	8/8	0.448
Safety-Gymnasium Goal (4 tasks)	3	6/8	0.264
Safety-Gymnasium Push and Button (5 tasks)	3	9/10	0.065
None (all 21 tasks)	3	37/42	0.361

Selection. The criterion selects $\lambda_{\max} = 3$ on 5,839 of the 5,985 subsets (97.6%) and $\lambda_{\max} = 1$ on the other 146 (Table 8). On the 17 selection tasks, the lead of $\lambda_{\max} = 3$ over the next best value has median $+0.016$, fifth percentile $+0.003$ and minimum -0.008. The next best value is $\lambda_{\max} = 1$ in 93% of the subsets. The minimum occurs when BallCircle, CarCircle, PointGoal1 and PointGoal2 are removed, and $\lambda_{\max} = 1$ is then selected. Holding out one of the four groups of tasks instead selects $\lambda_{\max} = 3$ in every case.

Held-out performance. On the held-out cells, the selected ceiling meets the budget on 7.0 of 8 cells on average and reaches a mean safe return of 0.358. Every task is held out equally often, so this mean equals the full-suite value 0.361 whenever $\lambda_{\max} = 3$ is selected. Its closeness to 0.361

therefore restates the selection frequency. It is not an estimate of how much the use of this suite during development inflates the reported results. Against the ceiling that would have been best on the held-out tasks in hindsight, the selected ceiling loses 0.013 in safe return on average (95th percentile 0.051, maximum 0.141).

Uncertainty over seeds. The lead of $\lambda_{\max} = 3$ in safe return is small. We resample the five seeds of each cell, with the same indices for every ceiling, and report 95% intervals over $2{,}000$ replicates; with five seeds these intervals are approximate. The difference in safe return is $+0.016$ $[-0.020, 0.052]$ against $\lambda_{\max} = 1$ and $+0.029$ $[-0.003, 0.059]$ against $\lambda_{\max} = 10$, so five seeds do not resolve either difference. The neighbouring values differ from 3 in coverage instead. $\lambda_{\max} = 1$ meets the budget on eight fewer cells (29 versus 37, interval $[4, 11]$). $\lambda_{\max} = 10$ meets it on three more (40), a difference within seed variation (interval $[-2, 5]$). The ceiling therefore sets an operating point, and the safe-return criterion places it at 3 for almost every subset of tasks.

Scope. These results describe stability within this suite and are not an independent validation. Any two of the 17-task subsets share at least 13 tasks, so the $5{,}985$ splits are not independent tests. Only the choice of the ceiling is held out. The noise prior, the actor averaging and the grid of candidate ceilings were fixed with all 21 tasks in view. All held-out tasks come from the groups used during development, so the analysis does not show how the choice transfers to new task families.

I Reproduction of the baselines

We run BC-Safe, BCQ-Lag, CPQ, COptiDICE and CDT from the OSRL implementation (Liu et al., 2024) with its official configurations, and FISOR and O3SRL from their authors' code. Every baseline number in this paper comes from these runs (§4), not from the original papers. This appendix records how the two relate.

What can be compared. A published number is directly comparable with ours only if the task and the cost limit coincide. Of the papers discussed here, only O3SRL reports the limit 40 on its own. DSRL (Liu et al., 2024) averages over three thresholds per task, so for it only returns can be compared. FISOR (Zheng et al., 2024) reports Safety-Gymnasium at limit 10 for five of our tasks (CarButton1/2, CarGoal1/2, CarPush2) and Bullet at limit 5; O3SRL (Chemingui et al., 2025a) reports PointCircle1/2 at limit 10 and four Bullet tasks at limit 40; FISOR re-reports BCQ-Lag, CPQ, COptiDICE and CDT at limit 10 on its five Car tasks, and CAPS (Chemingui et al., 2025b) reports CDT at limit 10 on all thirteen of our Safety-Gymnasium tasks. FISOR's paper also has a column "BC", a baseline that imitates the whole dataset rather than only the safe trajectories; its costs are two to five times those of our BC-Safe, and we do not use it as a reference for BC-Safe. For eight of our Safety-Gymnasium tasks (CarCircle1/2, PointCircle1/2, PointGoal1/2, PointPush1, PointButton1) we found no published number at limit 10 for BC-Safe, BCQ-Lag, CPQ or COptiDICE. For FISOR we found only O3SRL's report on PointCircle1/2. Apart from these two cells, our runs are the only reference for these five methods on those tasks. In total 41 cells are directly comparable (28 for the OSRL baselines, namely 5 each for BCQ-Lag, CPQ and COptiDICE from FISOR's paper and 13 for CDT from CAPS; 7 for FISOR, 5 from its own paper and 2 from O3SRL's report; 6 for O3SRL).

Agreement. Returns agree: no FISOR return differs from the published one by more than 0.10, CDT and BCQ-Lag agree in return on every directly comparable cell, and the 105 returns that can be set against DSRL's threshold averages show no significant difference. Costs differ more: CPQ differs on several cells that are unsafe in both sources, and our FISOR and O3SRL have the higher cost on a few cells. The safety verdict differs on the three cells of Table 9. With the published verdicts FISOR would meet the budget on 28 cells instead of 26 and O3SRL on 17 instead of 16, against 37 for C-Steering.

Why verdicts can differ. All three cells lie near the boundary and have a seed standard deviation of 0.5 to 2.0 budgets, so a mean over three seeds has a standard error of 0.3 to 1.2. Evaluation itself is noisy: re-running FISOR with the *same* seed changes the normalised cost of CarGoal1 from 2.19 to 1.48 and of CarPush2 from 0.57 to 0.81. On CarGoal1 the cost of FISOR also rises during training (from below 0.6 at 500k steps to 1.2–2.0 at 1M steps in the three runs for which we kept the history),

Table 9: The three cells, all at d=10, on which our re-run and the published number disagree on whether the budget is met. Cost is normalised by the budget; ours is mean $\pm$ std over 5 seeds, the published numbers use 3.

Method	Task	published cost	our cost	effect of using the published verdict
FISOR	CarGoal1	0.83	1.51 ± 0.54	FISOR gains a safe cell
FISOR	CarPush2	0.89	1.30 ± 0.70	FISOR gains a safe cell
O3SRL	PointCircle2	0.55	2.05 ± 2.01	O3SRL gains a safe cell

and the FISOR paper does not state which checkpoint it reports (O3SRL returns its last iterate); we report the final one for every method. For O3SRL on PointCircle1/2 a second, independent set of runs with the same seeds gave the same verdicts (2.66 on PointCircle2), and an independent report of FISOR on these two tasks (Chemingui et al., 2025a) agrees with ours (14.9 and 18.0 against our 15.8 and 19.2).

FISOR and the cost limit. FISOR learns a feasibility value by reachability analysis and its training does not involve the cost limit, which its code uses only to normalise the evaluated cost. Our two budgets are therefore two independently trained sets of the same model. Over the 105 task–seed pairs the mean absolute difference between the two sets is 0.030 in normalised return and 7.3 in raw episode cost, the size of the difference between two runs with the same seed (0.017 and 7.8).

Configurations. For our tasks the official configurations of BC-Safe, BCQ-Lag, CPQ and COptiDICE set only the episode length per task, and FISOR has a single configuration; CDT additionally sets per-task target returns and data-augmentation parameters. O3SRL's official code has entries for 10 of our 21 tasks (Bullet and PointCircle1/2), which set the episode length, the default cost limit and, for four Bullet tasks, a batch size of 1024 instead of 512; all algorithm hyper-parameters are shared. For the other 11 tasks we add entries of the same form (§4). O3SRL is safe on 15 of the 20 cells of the first group and on 1 of the 22 others.

J STEP-MATCHED BASELINES

The OSRL baselines are trained for their official 100k update steps and C-Steering for 500k. To check that the difference in training length does not drive the comparison, we re-trained BC-Safe and BCQ-Lag for 500k steps with otherwise identical official configurations (5 seeds, all 42 cells; Table 10). Neither changes materially: BC-Safe loses 2 safe cells and gains 0, BCQ-Lag loses 1 and gains 2, and the safe return of each stays within the bootstrap interval of its 100k version. The main tables report the official 100k runs.

Table 10: BC-Safe and BCQ-Lag at their official 100k update steps and at the 500k steps used by C-Steering. Columns as in Table 16.

Method	Budget met	met & R>0	Bullet	Safety-Gym.	mean return	excess cost	safe return
BC-Safe, 100k (main tables)	27/42	26/42	12/16	15/26	+0.325	0.499	0.211 [0.192, 0.228]
BC-Safe, 500k	25/42	24/42	11/16	14/26	+0.313	0.318	0.215 [0.198, 0.231]
BCQ-Lag, 100k (main tables)	6/42	6/42	3/16	3/26	+0.534	3.128	0.097 [0.083, 0.111]
BCQ-Lag, 500k	7/42	7/42	3/16	4/26	+0.535	3.132	0.085 [0.071, 0.100]

K PER-DECISION LATENCY

Table 11 reports the wall-clock time of one action selection for each method, measured on CarGoal1 (observation dimension 72) on a single NVIDIA GH200 with batch size one, including the transfer of the observation to the device and of the action back to the host: median over 500 consecutive decisions after 20 warm-up calls, using each method's own inference code and trained checkpoint (FISOR: the official implementation with its paper hyper-parameters; its latency does not depend on the weights). C-Steering and FISOR are JAX implementations; the OSRL baselines and O3SRL are PyTorch implementations, so absolute times are comparable within a framework and only indicative

across frameworks. The number of sequential network evaluations per decision is implementation-independent. All rows were timed one after another in a single job on the same GPU.

Table 11: Per-decision latency (median over 500 decisions, batch size 1, one GH200, all rows in one job).

Method	Networks evaluated per decision	Framework	Latency (ms)
C-Steering	noise actor, then one-step decoder (2 sequential passes)	JAX	0.45
FISOR	16 candidates $\times$ 5 denoising steps, then two critics on the 16	JAX	2.50
BC-Safe	one MLP	PyTorch	0.22
BCQ-Lag	CVAE decoder, then perturbation actor	PyTorch	0.43
O3SRL	one MLP (TD3+BC actor)	PyTorch	0.37
CDT	one transformer pass over a 10-step context	PyTorch	2.24

Sampling structure. Table 12 compares isolated sampling structures for C-Steering and ReFORM (Zhang et al., 2026), whose reflected flow generates the noise. Both are built from the same code with random weights and share the critic and the optimiser; this is not a timing comparison of the complete training algorithms. The ReFORM-style structure uses its published settings: 4×512 MLPs, ten flow steps, a one-step behaviour policy distilled from the ten-step flow, minibatches of 256 and gradient-norm clipping at 10. Both structures were timed in the same job as Table 11, with the observation already on the device. Per decision, ReFORM needs eleven sequential network passes against two. Its training update also integrates the behaviour flow for distillation and backpropagates through ten reflected steps, and takes about three times as long. With the noise flow reduced to the size of our actor, the update still takes 2.2 times as long, so most of the difference comes from the number of steps rather than the width of the networks. The complete C-Steering update, with its seven critics, dual step and weight averaging, takes 2.6 ms. On AntRun ($n = 8$) all times agree with CarGoal1 within 0.05 ms.

Table 12: Cost of the isolated sampling structures on CarGoal1: batch 1 per decision with the observation on the device, batch 256 per update, median over 500 decisions and 200 updates, one GH200. Parameters count the actor and behaviour-model networks used in the timed training update, including ReFORM's behaviour-flow teacher, but exclude the shared critic; they are not deployment parameter counts. Passes are sequential network evaluations per decision.

Structure	Passes	Decision (ms)	Update (ms)	Parameters
C-Steering (actor 2×256, one-step decoder)	2	0.23	1.09	0.91M
ReFORM, distilled policy	11	0.36	3.18	2.48M
ReFORM, ten-step behaviour flow	20	0.52	3.18	2.48M
ReFORM, noise flow 2×256	11	0.29	2.40	1.74M

L Reflected-flow actor: controlled comparison and diagnostics

This experiment separates the support of the noise prior from the mechanism used to choose noise within it. We compare C-Steering's direct actor with a ReFORM-style reflected-flow actor under the same constrained-learning configuration. All results in this appendix come from one set of runs with the protocol below.

L.1 Protocol and relation to ReFORM

We use OfflinePointGoal1Gymnasium-v0 with $n = 2$, $H = 1000$, budget $d = 10$, and seeds 100–102. Both variants use the full dataset, the projected-sphere prior of Eq. (7) with $R = \sqrt{3}$, the one-step MeanFlow decoder, reward and cost critics, amortised target, actor-loss normalisers, projected dual ascent with $\lambda_{\max} = 3$, and actor EMA of Table 7. Each run performs 500k updates and evaluates 20 episodes every 50k updates; we report the final checkpoint without selection. In this comparison both variants additionally clip the global gradient norm over all networks at 10,

before Adam. This clipping is absent from the main C-Steering configuration. These runs used a different machine from the main experiments (one NVIDIA RTX 4080, two runs sharing it) with the same package versions. With clipping on this machine, the direct actor has a mean C/d of 0.958 on seeds 100–102, against 0.79 for the main configuration on the same seeds, so the two actors are compared only within this experiment.

The direct actor is the 2×256 MLP with radial tanh of Section 3.3. The reflected actor is a 4×512 velocity-field MLP without layer normalisation, integrated for ten Euler steps from a fresh sample of the same projected-sphere prior on each call. We use the online actor in TD targets and the EMA actor at evaluation for both variants. The reflected actor is stochastic; the direct actor is deterministic. The actor comparison therefore includes differences in network size and stochasticity as well as the integration and compensation mechanism.

ReFORM uses a uniform ball prior, a behaviour flow trained by flow matching and a separately distilled one-step policy (Zhang et al., 2026). Here we retain C-Steering's prior, decoder and constrained objective to compare the noise actors on a common backbone. The ten integration steps, 4×512 velocity field and gradient-norm limit follow ReFORM, but these runs are not a reproduction of its complete algorithm. For the compensated branch we evaluate the right-hand side of Eq. (8), avoiding subtraction of two large terms when the proposed displacement greatly exceeds R. This is the same ideal update in exact arithmetic; the implementation uses a small epsilon in the normal direction. It removes the large-term cancellation of the literal expression, without bounding the velocity itself.

L.2 Final performance

Table 13: Final results of the controlled actor comparison. Each seed is evaluated over 20 episodes. The mean row is over seeds; the number of seeds meeting the budget is $2/3$ for the direct actor and $1/3$ for the reflected actor.

	Direct + clipping		Reflected flow + clipping	
Seed	C/d	R_{norm}	C/d	R_{norm}
100	0.905	0.653	1.210	0.347
101	0.885	0.631	1.490	0.368
102	1.085	0.700	0.925	0.371
Mean	0.958	0.661	1.208	0.362

Table 13 shows a higher final return for the direct actor at every seed and a lower cost at two of three seeds. Its mean cost meets the budget, while the reflected actor's mean exceeds it. Seed-level safe return is 0.428 and 0.124, respectively. These three-seed means are descriptive; they do not establish a statistically resolved safety difference. The reflected actor reaches $\lambda = 3$ at every recorded training point (every 5k updates), whereas the direct actor does so at 3% of recorded points and ends at 0.52, 1.39 and 0.53. This is a record of sampled training points, not of every optimiser step. Some earlier reflected-actor evaluations meet the budget, so the result concerns final performance rather than persistent infeasibility throughout training.

Run time. On the same GPU, a complete run of 500k updates with its ten evaluations took 27.0 minutes on average with the direct actor and 46.7 minutes with the reflected actor. Runs shared the GPU in pairs and the pairs differed, so these times are indicative; in the one pair that ran concurrently, the direct actor took 28.5 and the reflected actor 41.8 minutes. The isolated timing of Appendix K gives 1.1 against 3.2 ms per update for the two sampling structures; it does not measure complete training runs.

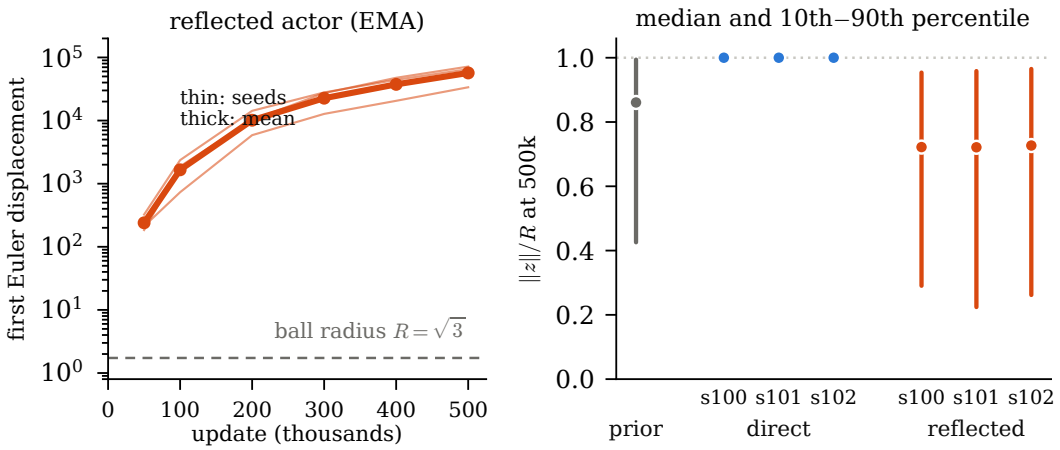


Figure 9: Reflected-flow diagnostics on PointGoal1, $d = 10$, three seeds. *Left:* mean first Euler displacement over 512 dataset states for each EMA actor (thin lines) and the mean over seeds (thick line); the dashed line is the ball radius. All tested Euler proposals at all ten steps exceed the radius at these checkpoints. *Right:* median and 10th–90th percentile of $\|z\|/R$ at the final checkpoint, shown separately for each seed; the prior uses 8192 samples. These are empirical percentile ranges, not confidence intervals.

L.3 What boundary compensation does

Write $\Delta_k = \Delta t\, v_\theta(s, z_k, k\Delta t)$, $\hat{z}_{k+1} = z_k + \Delta_k$, and $n_{k+1} = \hat{z}_{k+1}/\|\hat{z}_{k+1}\|$. On the branch $\|\hat{z}_{k+1}\| > R$, ReFORM's Eq. 12 gives

$$\begin{aligned} z_{k+1} &= \hat{z}_{k+1} - \langle \Delta_k, n_{k+1}\rangle n_{k+1} \\ &= \big(\langle \hat{z}_{k+1}, n_{k+1}\rangle - \langle \Delta_k, n_{k+1}\rangle\big) n_{k+1} = \langle z_k, n_{k+1}\rangle n_{k+1}. \end{aligned} \tag{10}$$

The second line uses that $\hat{z}_{k+1}$ is parallel to the unit vector n_{k+1}. By Cauchy–Schwarz, $\|z_{k+1}\| = |\langle z_k, n_{k+1}\rangle| \le \|z_k\|$. Thus, whenever all steps of an integration trigger compensation, the output radius cannot exceed the initial radius. Equality can hold, so a compensated step need not strictly shrink the radius. Uncompensated Euler steps can increase it. The result is a conditional property of this discrete update, not an impossibility result for reflected flows or for reaching the boundary. The direct radial actor avoids this sequence of compensated steps, although its tanh map can saturate near the boundary.

L.4 Checkpoint diagnostics

Diagnostics use 512 fixed dataset states and checkpoints at 50, 100, 200, 300, 400 and 500k updates. At each checkpoint we trace all ten Euler steps for both the online and EMA reflected actors. At 500k, the traced EMA output matches the agent's own generator exactly on the tested inputs at all three seeds. For the EMA actor, all tested proposals at every step, seed and diagnostic checkpoint lie outside the ball. Its mean first-step displacement grows from about 240 at 50k to 56,808 at 500k, against radius $R = \sqrt{3}$ (Figure 9). Clipping parameter gradients therefore does not, in these runs, prevent growth of the velocity-field output. This observation does not by itself identify why that output grows.

At the final checkpoint, the mean across seeds of the median radius $\|z\|/R$ is 0.723 for the EMA reflected actor, against approximately 1.000 for the direct actor and 0.861 for the sampled prior. Only 4.8% of reflected noise lies at $\|z\|/R > 0.98$, versus 94.3% for the direct actor; 61.0% lies below $0.8R$, versus 1.2%. The online reflected actor gives similar values (median 0.714, 4.2% near the boundary). No traced online or EMA sample exceeds $1.001R$ at any diagnostic checkpoint. Thus the reported performance gap persists with numerically stable compensation and without detected escape from the ball at this tolerance.

To examine variation over starting noise, we also draw 16 independent samples per state for the EMA reflected actor. At 500k the mean within-state coordinate standard deviations are 0.571 for

noise and 0.055 for decoded actions. Its cost ensemble (mean plus one standard deviation) predicts positive amortised cost for all 16 actions at 62.2% of dataset states; only 0.6% have some non-positive predictions while the mean prediction is positive. These are learned-critic diagnostics on dataset states, not environment safety labels, and they do not exclude an effect of stochasticity on performance.

Scope of the evidence. These runs show that the final gap persists when both actors use gradient clipping and the reflection is computed without cancellation. It also records repeated compensation and predominantly interior output noise, consistent with the non-increasing-radius property above. It does not establish that contraction causes the return or cost gap, that boundary noise is inherently safer, or that the original ReFORM algorithm fails on these tasks. We have not isolated network size, stochasticity, step count, velocity scale or compensation rule, and this comparison uses only $\lambda_{\max} = 3$. Mechanism-specific interventions and additional tasks are needed before making a broader claim.

M Further examples of the noise set

Figures 10–12 show, for one dataset state of three other tasks with two-dimensional actions (budget 10, seed 100), the sphere-prior pair of panels next to the ball-prior pair of Figure 1(b, c). In each figure the left pair is the noise set and the reachable actions of a decoder trained on the sphere prior, with the cross marking the noise chosen by that model's actor and its decoded action; the right pair is the ball prior of C-Steering, as in Figure 1(b, c).

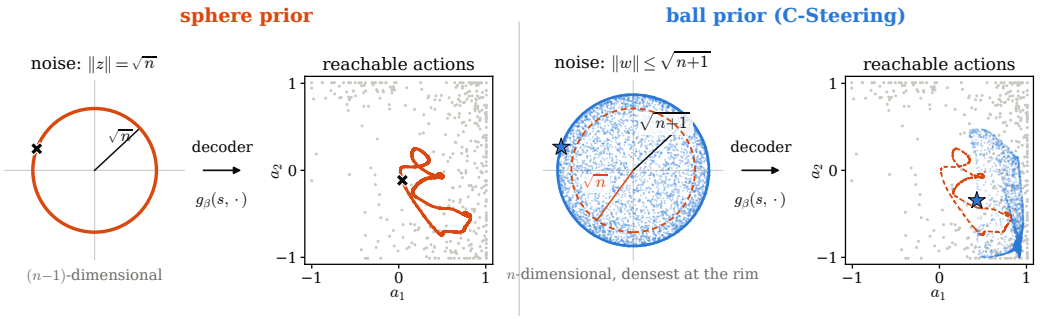


Figure 10: As Figure 1(b, c), for PointGoal1, with the sphere-prior pair on the left.

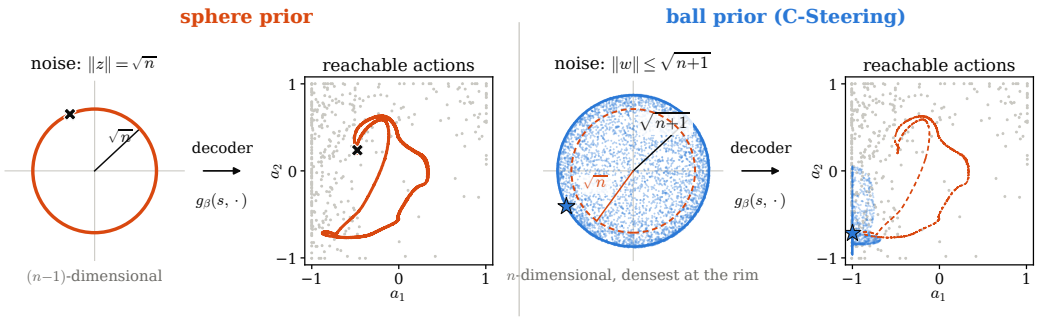


Figure 11: As Figure 1(b, c), for CarCircle1, with the sphere-prior pair on the left.

N Variants of the noise space

This appendix gives the per-task results behind block (e) of Table 6: the noise set and the evaluated actor are varied and everything else is the configuration of §3.4. Tables 14 and 15 list the cells

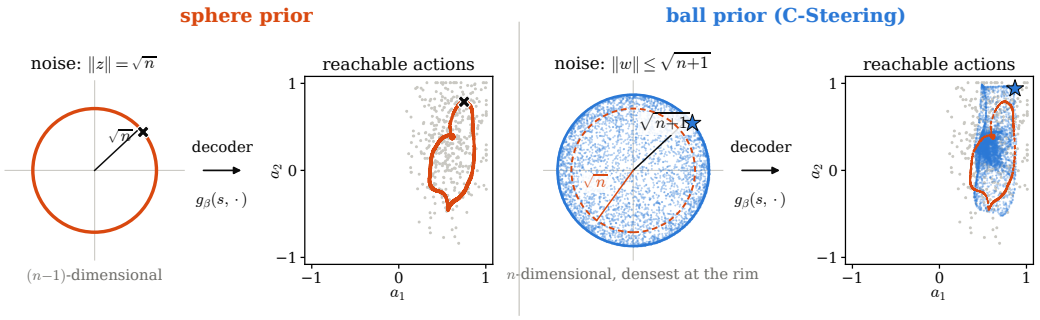


Figure 12: As Figure 1(b, c), for PointCircle1, with the sphere-prior pair on the left.

and Table 16 summarises them. The two ways of parameterising the actor on the full-dimensional prior behave alike; we use the radial map because the actor then has the dimension of the action. The trained actor places its noise within 2% of the boundary on 95% of dataset states, where the radial derivative of the $\tanh$ map is below 0.04. A radial projection, which is the identity inside the ball and does not saturate, does so on 97% of states and reaches 39 safe cells and a safe return of 0.370 (Table 16), against 37 and 0.361. The paired seed-bootstrap intervals of the two differences are $[-2, 5]$ cells and $[-0.018, 0.036]$. These results suggest that boundary selection is not explained solely by tanh saturation: the tested direct bounding maps give comparable outcomes. The separate reflected-flow comparison in Appendix L examines a multi-step generator.

O Uncertainty of the prior comparison and of the headline differences

This appendix quantifies the seed and task-composition uncertainty of the differences reported in Sections 4.1 and 4.2, from the per-seed results behind Tables 1, 4 and 6. Two bootstraps are used, each with $10{,}000$ resamples and a fixed seed. The *seed* bootstrap keeps all 42 cells and resamples the five seeds within each task, with the same seed indices for both budgets of a task; for the comparison of the two priors and of the action-space actor, which share seeds 100–104, the same indices are also used for both variants, so the difference is paired. Baselines are resampled independently, with their own seed counts. The *task* bootstrap resamples the 21 tasks with replacement, keeps both budgets of a selected task and does not resample seeds. The 21 tasks are a fixed benchmark rather than a sample from a population, so the task intervals describe sensitivity to the composition of this suite, not generalisation to new task families. The metrics are those of the main text: mean normalised return, the excess cost V, seed-level safe return and the number of cells meeting the budget.

The prior comparison. The return gain of the full-dimensional prior and its gain in seed-level safe return exclude zero under both bootstraps. The difference in excess cost does not. Its point estimate, $V = 0.056$ against 0.114, is dominated by three cells: CarButton2 at $d = 10$ (mean C/d 0.76 against 3.80 for the sphere) and DroneRun at $d = 10$ (0.10 against 2.14) favour the full-dimensional prior, while PointGoal2 at $d = 10$ (2.71 against 0.12) favours the sphere. Leaving out one task at a time moves the ratio of the two V values between 0.13 (without PointGoal2) and 1.18 (without CarButton2). The two priors meet the budget on the same number of cells but not on the same cells: CarButton1 at $d = 10$, CarButton2 at $d = 10$, DroneRun at $d = 10$ are met only with the full-dimensional prior, and CarPush2 at $d = 40$, CarRun at $d = 10$, PointGoal2 at $d = 10$ only with the sphere. Figure 14 shows every cell. The full-dimensional prior has the higher return on 28 of the 42 cells and the lower mean cost on 22.

Near-boundary cells. Six C-Steering cells have a mean cost within 10% of the budget: BallCircle at $d = 40$ (mean 0.92, seeds 0.90, 0.87, 0.90, 0.84, 1.08); CarGoal1 at $d = 40$ (mean 1.02, seeds 0.71, 1.12, 0.96, 1.12, 1.20); CarRun at $d = 10$ (mean 1.02, seeds 2.60, 0.68, 0.28, 1.10, 0.46); PointGoal1 at $d = 10$ (mean 0.91, seeds 0.64, 1.05, 0.68, 1.31, 0.85); PointGoal1 at $d = 40$ (mean 0.91, seeds 0.90, 0.87, 1.06, 0.89, 0.82); PointGoal2 at $d = 40$ (mean 0.94, seeds 0.91, 0.77,

Table 14: Variants of the noise space at cost limit d=10, same format as Table 1: normalised return and cost normalised by the budget, mean ± std over 5 seeds; grey misses the budget, blue is the best safe return in the row. "sphere" trains the decoder on the sphere of radius $\sqrt{n}$ and confines the actor to it; "ball" and "disc" train the decoder on the full-dimensional prior of Eq. (7), and differ in the actor, which outputs a point of the sphere one dimension up whose last coordinate is dropped (ball) or a point of the ball through the radial $\tanh$ map (disc); EMA evaluates an exponential moving average of the actor's weights. "disc + EMA" is C-Steering, the configuration of the main tables.

Task		sphere	sphere + EMA	ball	disc	ball + EMA	**disc + EMA**
Bullet-Safety-Gym							
BallCircle	reward ↑	0.72 ± 0.04	0.71 ± 0.02	0.75 ± 0.02	0.76 ± 0.01	0.76 ± 0.01	0.76 ± 0.01
	cost ↓	0.74 ± 0.80	0.43 ± 0.35	0.33 ± 0.43	0.25 ± 0.26	0.19 ± 0.09	0.21 ± 0.20
BallRun	reward ↑	0.12 ± 0.09	0.10 ± 0.09	0.08 ± 0.11	0.09 ± 0.07	0.09 ± 0.07	0.08 ± 0.09
	cost ↓	1.00 ± 1.90	0.12 ± 0.24	0.26 ± 0.38	0.45 ± 0.90	0.00 ± 0.00	0.52 ± 1.04
CarCircle	reward ↑	0.69 ± 0.02	0.71 ± 0.02	0.74 ± 0.01	0.74 ± 0.01	0.75 ± 0.00	0.76 ± 0.01
	cost ↓	0.42 ± 0.34	0.18 ± 0.19	0.41 ± 0.49	0.13 ± 0.07	0.18 ± 0.12	0.15 ± 0.09
CarRun	reward ↑	0.97 ± 0.01	0.96 ± 0.01	0.97 ± 0.01	0.97 ± 0.00	0.97 ± 0.01	0.97 ± 0.00
	cost ↓	1.63 ± 1.21	0.78 ± 1.14	0.99 ± 0.33	0.86 ± 0.63	1.03 ± 0.57	1.02 ± 0.84
DroneCircle	reward ↑	0.53 ± 0.02	0.54 ± 0.01	0.56 ± 0.02	0.57 ± 0.01	0.58 ± 0.01	0.58 ± 0.01
	cost ↓	0.02 ± 0.04	0.01 ± 0.01	0.01 ± 0.02	0.01 ± 0.01	0.00 ± 0.00	0.01 ± 0.02
DroneRun	reward ↑	0.22 ± 0.19	0.30 ± 0.18	0.46 ± 0.11	0.56 ± 0.02	0.51 ± 0.08	0.57 ± 0.01
	cost ↓	1.36 ± 2.71	2.14 ± 3.10	3.55 ± 2.84	0.63 ± 1.15	2.25 ± 1.95	0.10 ± 0.19
AntCircle	reward ↑	0.56 ± 0.03	0.55 ± 0.04	0.54 ± 0.02	0.56 ± 0.02	0.54 ± 0.03	0.56 ± 0.04
	cost ↓	0.01 ± 0.01	0.00 ± 0.00	0.01 ± 0.01	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
AntRun	reward ↑	0.67 ± 0.03	0.67 ± 0.03	0.62 ± 0.05	0.61 ± 0.07	0.62 ± 0.06	0.62 ± 0.06
	cost ↓	0.78 ± 0.51	0.75 ± 0.55	0.76 ± 0.34	0.72 ± 0.53	0.94 ± 0.50	0.71 ± 0.42
Safety-Gymnasium							
CarCircle1	reward ↑	0.44 ± 0.08	0.51 ± 0.05	0.53 ± 0.05	0.57 ± 0.04	0.54 ± 0.03	0.56 ± 0.03
	cost ↓	0.84 ± 0.65	0.97 ± 0.94	0.50 ± 0.32	0.63 ± 0.96	0.83 ± 0.97	0.61 ± 0.59
CarCircle2	reward ↑	0.27 ± 0.08	0.31 ± 0.10	0.45 ± 0.01	0.49 ± 0.02	0.49 ± 0.03	0.49 ± 0.02
	cost ↓	0.00 ± 0.00	0.03 ± 0.05	0.86 ± 0.84	0.52 ± 0.56	0.73 ± 0.46	0.65 ± 0.69
PointCircle1	reward ↑	0.34 ± 0.05	0.38 ± 0.04	0.46 ± 0.01	0.47 ± 0.01	0.46 ± 0.01	0.47 ± 0.01
	cost ↓	0.33 ± 0.58	0.11 ± 0.13	0.02 ± 0.04	0.20 ± 0.25	0.21 ± 0.42	0.00 ± 0.00
PointCircle2	reward ↑	0.29 ± 0.08	0.35 ± 0.11	0.38 ± 0.05	0.40 ± 0.03	0.37 ± 0.04	0.41 ± 0.03
	cost ↓	0.07 ± 0.11	0.33 ± 0.65	0.00 ± 0.00	0.04 ± 0.05	0.00 ± 0.00	0.01 ± 0.02
CarGoal1	reward ↑	0.04 ± 0.01	0.03 ± 0.01	0.01 ± 0.02	0.01 ± 0.02	0.01 ± 0.02	0.00 ± 0.02
	cost ↓	0.88 ± 0.83	0.25 ± 0.32	0.04 ± 0.05	0.11 ± 0.11	0.41 ± 0.76	0.10 ± 0.07
CarGoal2	reward ↑	0.12 ± 0.03	0.11 ± 0.05	0.06 ± 0.02	0.07 ± 0.03	0.09 ± 0.04	0.07 ± 0.02
	cost ↓	0.89 ± 0.89	0.55 ± 0.43	0.46 ± 0.32	1.16 ± 1.82	0.76 ± 0.81	0.54 ± 0.39
PointGoal1	reward ↑	0.43 ± 0.04	0.46 ± 0.04	0.65 ± 0.01	0.63 ± 0.02	0.67 ± 0.02	0.64 ± 0.03
	cost ↓	0.41 ± 0.24	0.41 ± 0.23	1.23 ± 0.45	1.22 ± 0.39	0.98 ± 0.38	0.91 ± 0.25
PointGoal2	reward ↑	0.12 ± 0.04	0.12 ± 0.05	0.09 ± 0.04	0.11 ± 0.07	0.08 ± 0.04	0.11 ± 0.02
	cost ↓	0.25 ± 0.27	0.12 ± 0.07	1.99 ± 1.31	1.79 ± 0.99	0.97 ± 1.01	2.71 ± 2.74
CarPush2	reward ↑	-0.02 ± 0.04	-0.02 ± 0.03	-0.01 ± 0.03	-0.03 ± 0.03	0.01 ± 0.04	-0.03 ± 0.03
	cost ↓	0.07 ± 0.09	0.06 ± 0.08	0.10 ± 0.08	0.02 ± 0.02	0.05 ± 0.03	0.09 ± 0.11
PointPush1	reward ↑	0.00 ± 0.02	-0.00 ± 0.02	0.05 ± 0.04	0.11 ± 0.04	0.07 ± 0.04	0.10 ± 0.07
	cost ↓	0.13 ± 0.26	0.00 ± 0.00	0.11 ± 0.10	0.12 ± 0.11	0.06 ± 0.08	0.14 ± 0.24
CarButton1	reward ↑	0.02 ± 0.03	0.01 ± 0.01	-0.00 ± 0.01	-0.02 ± 0.02	-0.01 ± 0.01	-0.01 ± 0.01
	cost ↓	1.95 ± 1.17	1.29 ± 0.90	0.48 ± 0.32	0.33 ± 0.17	0.21 ± 0.18	0.50 ± 0.30
CarButton2	reward ↑	0.15 ± 0.12	0.10 ± 0.08	-0.01 ± 0.01	-0.00 ± 0.01	-0.01 ± 0.01	-0.01 ± 0.01
	cost ↓	5.61 ± 4.81	3.80 ± 2.28	0.57 ± 0.46	0.38 ± 0.14	0.46 ± 0.32	0.76 ± 0.29
PointButton1	reward ↑	-0.05 ± 0.04	-0.06 ± 0.05	-0.08 ± 0.05	-0.15 ± 0.08	-0.08 ± 0.06	-0.16 ± 0.05
	cost ↓	0.38 ± 0.24	0.38 ± 0.03	0.49 ± 0.27	0.26 ± 0.09	0.41 ± 0.26	0.17 ± 0.09
Average	reward ↑	0.32	0.33	0.35	0.36	0.36	0.36
	cost ↓	0.85	0.61	0.63	0.47	0.51	0.47
Budget met		16/21	18/21	18/21	18/21	19/21	19/21

1.15, 1.12, 0.74). Their seed values show that a cell's label can change with one seed; the budget-multiplier counts in Table 5 and the seed-level metrics are the corresponding checks. Cells farther from the boundary are not thereby certain; the seed spread of every cell is in Tables 1 and 4.

Table 15: Variants of the noise space at cost limit d=40, same format as Table 4: normalised return and cost normalised by the budget, mean ± std over 5 seeds; grey misses the budget, blue is the best safe return in the row. "sphere" trains the decoder on the sphere of radius $\sqrt{n}$ and confines the actor to it; "ball" and "disc" train the decoder on the full-dimensional prior of Eq. (7), and differ in the actor, which outputs a point of the sphere one dimension up whose last coordinate is dropped (ball) or a point of the ball through the radial $\tanh$ map (disc); EMA evaluates an exponential moving average of the actor's weights. "disc + EMA" is C-Steering, the configuration of the main tables.

Task		sphere	sphere + EMA	ball	disc	ball + EMA	**disc + EMA**
Bullet-Safety-Gym							
BallCircle	reward ↑	0.86 ± 0.03	0.84 ± 0.02	0.87 ± 0.03	0.87 ± 0.05	0.89 ± 0.02	0.89 ± 0.02
	cost ↓	1.08 ± 0.32	0.91 ± 0.13	0.91 ± 0.16	0.89 ± 0.30	0.95 ± 0.12	0.92 ± 0.08
BallRun	reward ↑	0.02 ± 0.12	0.04 ± 0.11	0.10 ± 0.11	0.05 ± 0.11	0.10 ± 0.06	0.04 ± 0.09
	cost ↓	0.06 ± 0.11	0.13 ± 0.27	0.27 ± 0.54	0.13 ± 0.27	0.00 ± 0.00	0.00 ± 0.00
CarCircle	reward ↑	0.71 ± 0.07	0.73 ± 0.01	0.75 ± 0.04	0.75 ± 0.05	0.75 ± 0.04	0.78 ± 0.03
	cost ↓	0.52 ± 0.67	0.30 ± 0.13	0.75 ± 0.41	0.53 ± 0.55	0.43 ± 0.39	0.65 ± 0.47
CarRun	reward ↑	0.99 ± 0.04	0.97 ± 0.01	0.99 ± 0.01	0.98 ± 0.01	0.99 ± 0.01	0.98 ± 0.01
	cost ↓	1.62 ± 1.78	0.58 ± 0.67	0.94 ± 0.59	0.76 ± 0.43	0.65 ± 0.40	0.52 ± 0.32
DroneCircle	reward ↑	0.66 ± 0.11	0.65 ± 0.11	0.61 ± 0.14	0.67 ± 0.14	0.62 ± 0.07	0.64 ± 0.07
	cost ↓	0.80 ± 0.60	0.64 ± 0.58	0.58 ± 0.77	0.91 ± 0.84	0.46 ± 0.38	0.65 ± 0.62
DroneRun	reward ↑	0.62 ± 0.04	0.58 ± 0.06	0.62 ± 0.05	0.60 ± 0.07	0.63 ± 0.08	0.66 ± 0.07
	cost ↓	1.33 ± 0.44	1.53 ± 0.58	1.35 ± 0.41	1.29 ± 0.61	1.12 ± 0.55	1.31 ± 0.61
AntCircle	reward ↑	0.79 ± 0.08	0.80 ± 0.04	0.78 ± 0.07	0.77 ± 0.07	0.81 ± 0.04	0.81 ± 0.04
	cost ↓	0.93 ± 0.98	0.35 ± 0.08	0.75 ± 0.78	0.74 ± 0.70	0.51 ± 0.09	0.53 ± 0.18
AntRun	reward ↑	0.69 ± 0.02	0.69 ± 0.02	0.68 ± 0.03	0.68 ± 0.02	0.69 ± 0.02	0.69 ± 0.02
	cost ↓	0.42 ± 0.11	0.43 ± 0.12	0.46 ± 0.08	0.49 ± 0.14	0.52 ± 0.08	0.50 ± 0.18
Safety-Gymnasium							
CarCircle1	reward ↑	0.58 ± 0.10	0.58 ± 0.08	0.55 ± 0.05	0.56 ± 0.04	0.53 ± 0.04	0.55 ± 0.03
	cost ↓	0.70 ± 0.46	0.56 ± 0.43	0.27 ± 0.36	0.12 ± 0.10	0.10 ± 0.12	0.14 ± 0.15
CarCircle2	reward ↑	0.31 ± 0.07	0.34 ± 0.05	0.48 ± 0.02	0.50 ± 0.03	0.48 ± 0.02	0.52 ± 0.02
	cost ↓	0.01 ± 0.02	0.00 ± 0.00	0.07 ± 0.07	0.20 ± 0.27	0.10 ± 0.14	0.25 ± 0.20
PointCircle1	reward ↑	0.38 ± 0.06	0.36 ± 0.06	0.47 ± 0.04	0.49 ± 0.03	0.46 ± 0.03	0.48 ± 0.02
	cost ↓	0.34 ± 0.46	0.40 ± 0.48	0.12 ± 0.21	0.00 ± 0.00	0.24 ± 0.26	0.07 ± 0.09
PointCircle2	reward ↑	0.36 ± 0.07	0.35 ± 0.06	0.40 ± 0.04	0.42 ± 0.04	0.41 ± 0.05	0.43 ± 0.04
	cost ↓	0.01 ± 0.02	0.05 ± 0.11	0.02 ± 0.03	0.00 ± 0.00	0.02 ± 0.02	0.00 ± 0.00
CarGoal1	reward ↑	0.69 ± 0.04	0.70 ± 0.02	0.75 ± 0.03	0.75 ± 0.02	0.75 ± 0.02	0.75 ± 0.02
	cost ↓	0.82 ± 0.25	1.02 ± 0.22	1.01 ± 0.14	0.87 ± 0.05	0.99 ± 0.18	1.02 ± 0.18
CarGoal2	reward ↑	0.25 ± 0.05	0.26 ± 0.07	0.32 ± 0.05	0.28 ± 0.06	0.28 ± 0.05	0.30 ± 0.03
	cost ↓	0.82 ± 0.60	0.54 ± 0.31	0.65 ± 0.29	0.97 ± 0.19	0.81 ± 0.28	0.70 ± 0.15
PointGoal1	reward ↑	0.76 ± 0.02	0.77 ± 0.02	0.78 ± 0.01	0.78 ± 0.01	0.78 ± 0.02	0.78 ± 0.01
	cost ↓	0.88 ± 0.12	0.90 ± 0.08	0.73 ± 0.16	0.96 ± 0.15	0.92 ± 0.17	0.91 ± 0.08
PointGoal2	reward ↑	0.51 ± 0.12	0.49 ± 0.06	0.59 ± 0.11	0.60 ± 0.08	0.63 ± 0.04	0.65 ± 0.09
	cost ↓	0.74 ± 0.28	0.72 ± 0.45	1.04 ± 0.16	1.01 ± 0.28	0.92 ± 0.09	0.94 ± 0.17
CarPush2	reward ↑	0.12 ± 0.04	0.12 ± 0.05	0.29 ± 0.05	0.25 ± 0.09	0.31 ± 0.06	0.28 ± 0.03
	cost ↓	0.26 ± 0.25	0.63 ± 0.63	1.26 ± 0.60	0.80 ± 0.23	1.05 ± 0.24	1.27 ± 0.63
PointPush1	reward ↑	0.16 ± 0.07	0.15 ± 0.06	0.31 ± 0.11	0.39 ± 0.08	0.29 ± 0.08	0.39 ± 0.09
	cost ↓	0.24 ± 0.11	0.61 ± 0.78	0.57 ± 0.25	0.46 ± 0.10	0.50 ± 0.27	0.39 ± 0.19
CarButton1	reward ↑	-0.01 ± 0.01	-0.00 ± 0.00	0.00 ± 0.01	-0.01 ± 0.01	-0.00 ± 0.01	-0.01 ± 0.01
	cost ↓	0.14 ± 0.05	0.09 ± 0.05	0.18 ± 0.15	0.10 ± 0.07	0.26 ± 0.23	0.10 ± 0.07
CarButton2	reward ↑	0.01 ± 0.02	0.01 ± 0.01	-0.03 ± 0.01	-0.02 ± 0.01	-0.02 ± 0.01	-0.01 ± 0.01
	cost ↓	0.21 ± 0.10	0.16 ± 0.06	0.06 ± 0.05	0.06 ± 0.02	0.21 ± 0.24	0.04 ± 0.03
PointButton1	reward ↑	0.02 ± 0.02	-0.01 ± 0.05	0.01 ± 0.03	0.04 ± 0.03	0.07 ± 0.04	0.03 ± 0.03
	cost ↓	0.16 ± 0.06	0.23 ± 0.13	0.22 ± 0.08	0.16 ± 0.08	0.25 ± 0.12	0.18 ± 0.03
Average	reward ↑	0.45	0.45	0.49	0.50	0.50	0.50
	cost ↓	0.58	0.51	0.58	0.55	0.53	0.53
Budget met		18/21	19/21	17/21	19/21	19/21	18/21

Baselines. Against BC-Safe and FISOR, the differences in excess cost, safe return and safe cells exclude zero under both bootstraps; the return difference against FISOR touches zero under the task bootstrap. Against the action-space actor at $\alpha = 0.1$, C-Steering has lower return and the same safe return within seed variation, and meets the budget on more cells with lower excess cost.

Table 16: Summary of the noise-space variants over all 42 cells. "Excess cost" is the mean over cells of $\max(0, C/d - 1)$; "safe return" is the mean over cells and seeds of the normalised return of runs that meet the budget (unsafe runs count as zero), with a 95% bootstrap interval over seeds.

Variant	Budget met	met & R>0	Bullet	Safety-Gymnasium	mean return	excess cost	safe return	runs that meet the budget
sphere	34/42	31/42	10/16	24/26	+0.384	0.181	0.277 [0.250, 0.303]	78%
sphere + EMA	37/42	32/42	14/16	23/26	+0.387	0.114	0.324 [0.304, 0.344]	85%
ball	35/42	30/42	14/16	21/26	+0.420	0.106	0.315 [0.288, 0.342]	81%
disc	37/42	31/42	15/16	22/26	+0.426	0.035	0.342 [0.317, 0.366]	84%
ball + EMA	38/42	33/42	13/16	25/26	+0.428	0.035	0.348 [0.322, 0.373]	86%
disc + EMA	37/42	31/42	14/16	23/26	+0.432	0.056	0.361 [0.338, 0.383]	86%
disc + EMA, radial projection	39/42	34/42	15/16	24/26	+0.416	0.021	0.370 [0.350, 0.389]	89%

Table 17: Differences with 95% bootstrap intervals. For each metric the first column gives the point estimate and the seed-bootstrap interval, the second the task-bootstrap interval. The action-space actor shares seeds with C-Steering but was not included in the task bootstrap.

	Δ mean return		ΔV		Δ safe return		Δ safe cells	
Comparison	seeds	tasks	seeds	tasks	seeds	tasks	seeds	tasks
full-dim. prior − sphere prior	+0.045 [+0.033, +0.057]	[+0.017, +0.075]	−0.058 [−0.169, +0.035]	[−0.238, +0.090]	+0.036 [+0.007, +0.066]	[+0.002, +0.074]	+0 [−3, +4]	[−5, +5]
C-Steering − action actor, α=0.1	−0.024 [−0.031, −0.017]	—	−0.223 [−0.354, −0.107]	—	−0.005 [−0.032, +0.021]	—	+6 [+2, +8]	—
C-Steering − BC-Safe	+0.107 [+0.097, +0.118]	[+0.043, +0.172]	−0.444 [−0.574, −0.329]	[−0.962, −0.089]	+0.150 [+0.121, +0.179]	[+0.080, +0.223]	+10 [+7, +14]	[+2, +18]
C-Steering − FISOR	+0.082 [+0.074, +0.091]	[−0.000, +0.175]	−1.581 [−1.706, −1.450]	[−2.925, −0.371]	+0.185 [+0.161, +0.210]	[+0.099, +0.274]	+11 [+7, +14]	[+2, +20]

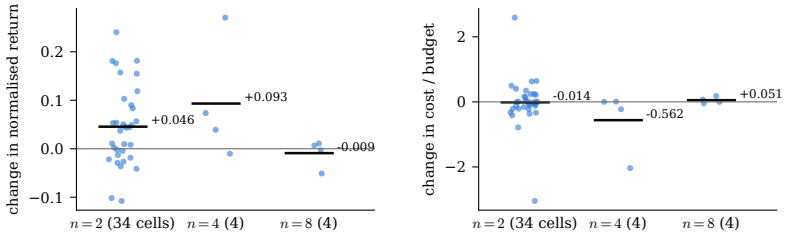


Figure 13: Effect of replacing spherical noise with bounded full-dimensional noise, grouped by action dimension. Both variants evaluate an averaged actor. Group sizes are 34 cells at $n = 2$, four at $n = 4$, and four at $n = 8$; the per-cell differences are shown in the figure.

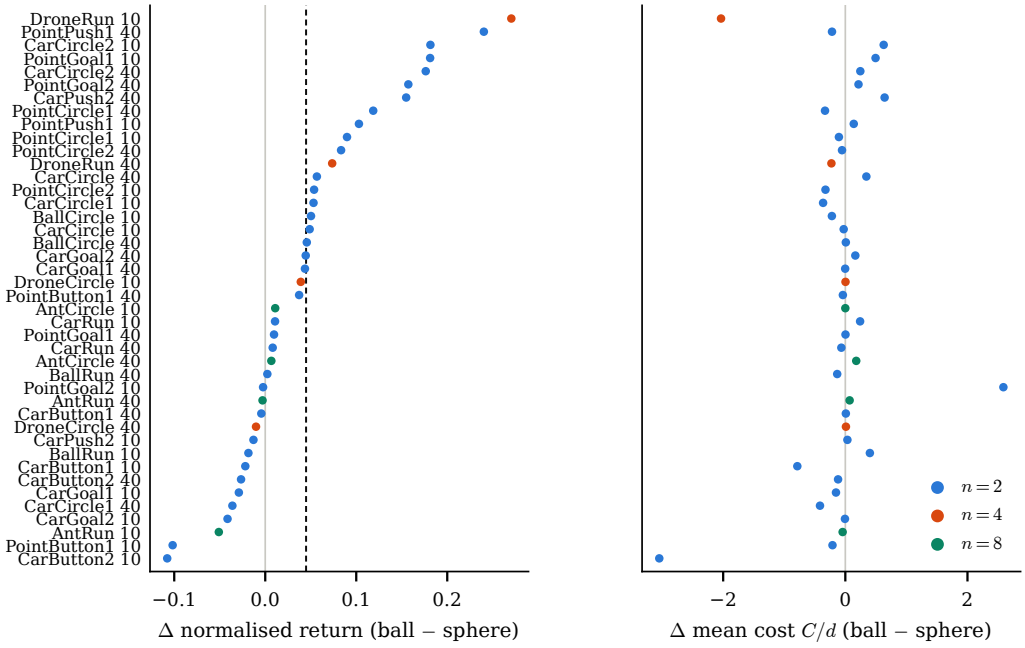


Figure 14: Per-cell differences between the full-dimensional prior (C-Steering) and the sphere prior, both with the averaged actor: normalised return (left, dashed line: mean over cells) and mean cost C/d (right), sorted by the return difference and coloured by action dimension.

P BOUNDARY PROJECTION AND DECODER GEOMETRY DIAGNOSTICS

These diagnostics extend the five-seed performance comparison in Appendix O by examining where the actor selects noise and how the learned decoder maps that region into actions. Three findings characterise the ball configuration: actor outputs concentrate near the boundary, projecting them to the exact boundary rarely changes critic feasibility labels, and the two-dimensional decoded boundary image has approximately 19% larger RMS spread. We also examine angular uniformity and feasible-region fragmentation to distinguish these observations from alternative geometric explanations. Throughout this appendix, "ball" denotes C-Steering (the disc + EMA configuration), and "sphere" denotes the sphere + EMA configuration.

Protocol and scope. We use final checkpoints for seeds 100–102 on all 42 task–budget cells. The projection diagnostic uses 128 dataset states per cell and seed; the boundary-section diagnostic uses 64 states and 120 angular samples per section, giving 3° spacing. Directions are shared across decoder comparisons and multiplied by each prior's own radius: $\sqrt{n+1}$ for ball and $\sqrt{n}$ for sphere. For $n = 2$ this traces the full noise boundary; for higher dimensions a circular section is only a subset of that boundary. The principal geometric comparison is therefore the 34 cells with $n = 2$; the four cells at each of $n = 4$ and $n = 8$ are descriptive checks. Reported geometric confidence intervals use task-cluster bootstrap resampling, preserving the two budgets of each task. They describe sensitivity within this benchmark, not uncertainty over new task families. All feasibility labels in this appendix are predictions of learned cost critics at dataset states, not observed episodic safety.

Projecting selected noise onto the boundary. For nonzero actor noise w, define $w_\partial = Rw/\|w\|$ and the root-mean-square action change

$$\Delta a(s) = \frac{\|g_\beta(s, w) - g_\beta(s, w_\partial)\|_2}{\sqrt{n}}. \tag{11}$$

Table 18 reports the near-boundary share, action-change summaries and critic-label changes. "Own" scores the ball actions with its own cost critic; "cross" uses the paired sphere critic. The feasible fractions for actor and projected actions in the table use the own critic.

Table 18: Boundary projection on dataset states, three seeds. Near-boundary means $\|w\|/R > 0.98$. Action changes use Eq. (11). Flip rates and feasible fractions are proportions. Action-change quantiles are taken over states for each checkpoint and then averaged over seeds and cells. Small label changes do not certify unchanged rollout performance.

Dimension (cells)	Near-boundary	Δa median	Δa p90	Δa p95	Label flips own / cross	Feasible actor / projected
$n = 2$ (34)	0.94	0.0000	0.035	0.094	0.002 / 0.001	0.59 / 0.59
$n = 4$ (4)	0.98	0.0000	0.007	0.024	0.000 / 0.000	0.62 / 0.62
$n = 8$ (4)	0.92	0.0005	0.017	0.031	0.000 / 0.000	0.61 / 0.61

The near-boundary share in this three-seed diagnostic is consistent with the original approximately 95% observation. Projection rarely changes the predicted feasible label, and the own-critic feasible fractions agree at the reported precision. Nevertheless, the action-change tail is not uniformly small: the $n = 2$ p90 is 0.035, above a threshold of 0.02 that we fixed before running the diagnostic, and $14/42$ cells exceed it. We therefore also evaluate sections at the ball actor's actual radius. These checks support exact-boundary sections as a local diagnostic of critic labels, but do not show that interior access is dispensable. No environment rollout of the projected policy was performed; small one-step action changes can alter subsequent states and episodic outcomes.

Proximity to logged actions. Using the local-support diagnostic of Appendix B, selected ball actions are somewhat less often flagged outside nearby logged actions than selected sphere actions, but this difference is much smaller than the random-noise comparison in Figure 1d. At $n = 2$, selected ball actions also lie farther from the local action mean in units of local spread (Table 19). Distance from a local mean and proximity to individual logged actions measure different properties; neither certifies safety or exact conditional data support.

Geometry of the boundary image. Table 20 summarises the principal two-dimensional results. G1 is the diagnostic's p90-to-median angular unevenness statistic; it measures nonuniformity along

Table 19: Local-support diagnostics for selected and boundary-projected actions. Outside-local-support rates are percentages; the local-mean distance is the dimensionless typicality score defined in Appendix B. Each triple orders ball actor / projected ball actor / sphere actor.

Dimension (cells)	Outside local support (%)	Local-mean distance
$n = 2$ (34)	14.0 / 15.2 / 17.6	1.02 / 1.05 / 0.88
$n = 4$ (4)	16.2 / 17.0 / 22.5	0.92 / 0.93 / 0.93
$n = 8$ (4)	12.6 / 12.8 / 13.2	0.78 / 0.79 / 0.77

the sampled parameterisation, not smoothness in every geometric sense. G2 measures the RMS spread of the decoded boundary image. The ball decoder has slightly higher angular unevenness ($+0.19$, interval $[+0.01, +0.39]$) and a larger image spread (0.56 versus 0.47, approximately 19% larger). Arc length has no clear difference. Thus the measured unevenness does not support a more uniform ball boundary. A larger image spread is an observed difference, not an explanation of improved return. No noise-ignoring decoder collapse was detected by the diagnostic.

Table 20: Boundary-section diagnostics on the 34 cells with $n = 2$, three seeds. Differences are ball minus sphere, with 95% task-cluster bootstrap intervals. Own scoring uses each decoder's associated critic; cross scoring exchanges the paired critics. Feasible fractions and angular margins are critic-derived diagnostics. Rounded zero endpoints should not be interpreted as decisive evidence.

Metric	Ball	Sphere	Difference [95% interval]
G1 angular unevenness (p90/median)	2.80	2.61	$+0.19$ $[+0.01, +0.39]$
Boundary-image arc length	3.98	4.22	-0.23 $[-0.65, +0.21]$
G2 boundary-image RMS spread	0.56	0.47	$+0.09$ $[+0.04, +0.14]$
Q1 feasible fraction, own	0.585	0.556	$+0.03$ $[+0.00, +0.06]$
Q1 feasible fraction, cross	0.556	0.584	-0.03 $[-0.06, +0.00]$
Q2 feasible arc count, own	0.61	0.58	$+0.04$ $[+0.01, +0.06]$
Q3 angular margin, own	137°	125°	$+12.7^\circ$ $[-3.9, +29.3]$
Q3 angular margin, cross	126°	138°	-12.0° $[-28.5, +4.0]$
Q3 own, ball at actor's actual radius	139°	125°	$+13.7^\circ$ $[-2.6, +29.8]$

The supplementary far-angle near-neighbour statistic G3 is 0.133 versus 0.135 at action-distance threshold $\epsilon = 0.05$ (difference interval $[-0.02, +0.02]$), and 0.23 versus 0.26 at $\epsilon = 0.10$ (interval $[-0.07, -0.00]$, rounded). These absolute-distance comparisons are sensitive to image scale: the larger ball image can reduce near-neighbour rates without less folding. We therefore do not treat the coarse-threshold difference as evidence of superior boundary geometry.

Critic-predicted feasibility and fragmentation. For each section, Q1 measures the fraction labelled feasible, Q2 counts feasible arcs, and Q3 summarises angular feasibility margin around the actor's point. With own critics, the ball has a slightly larger feasible fraction; exchanging critics reverses the difference. Angular-margin differences also reverse and their intervals include zero. This sensitivity prevents attribution of a feasibility advantage to the decoder alone. It does not establish that critic error alone causes the difference: the paired critics evaluate different learned policies, and cross scoring may introduce additional evaluation error.

Only about 3% of queried states have both feasible and infeasible points on the section. For ball and sphere, approximately 57% and 54% are feasible throughout, while 40% and 43% are infeasible throughout. Consequently, the aggregate feasible-arc counts largely reflect these all-or-none states. On mixed states both decoders have about 1.2–1.5 feasible arcs. The margin distribution is likewise dominated by endpoints: at roughly 41–45% of states the actor point is itself infeasible (zero margin), while approximately 53–57% have fully feasible sections (censored margin). The remaining states have own-critic median margins of 60° for both decoders. These data provide little support for fragmentation as an explanation and limited information about margin differences on mixed states; failure to detect a difference is not evidence of equivalence.

Using the ball actor's actual radius instead of the exact boundary produces a similar own-critic margin comparison (Table 20). At $n = 8$, angular unevenness reverses direction (ball 1.98, sphere 2.25), while own and cross margin differences are $+18°$ [0, 36] and $-14°$ [-28, 0], respectively. These higher-dimensional results remain descriptive because of the small number of cells and restricted sections.

Interpretation and missing interventions. The measured two-dimensional angular unevenness runs against the hypothesised more uniform ball boundary, and the feasibility diagnostics do not support a critic-independent advantage in connectedness or margin. The performance gain remains present in the separate five-seed analysis, but these measurements do not explain it. They are limited to final checkpoints, three seeds, dataset-state queries and discretised sections; the jointly trained actor–critic systems and differing prior radii prevent causal isolation. Radial training profiles and coverage over training were not measured. Evaluating boundary-projected policies in full episodes would test deployment sensitivity; retraining a sphere at the ball's radius would probe radius sensitivity; and training a boundary-restricted actor with the ball prior would test the role of interior access during optimisation. These interventions have not been performed, and no internal-support regularisation mechanism is claimed from the current evidence.

Q PER-CELL HEAD-TO-HEAD

Table 21 compares C-Steering with each baseline on every task–budget cell, using the cell means of Tables 1–4. Where both methods meet the budget the entry is the difference in normalised return (C-Steering minus baseline); otherwise it records which of the two is safe. Return differences are not reported for cells on which either method is unsafe, since such returns are not comparable.

Table 21: C-Steering against each baseline, per cell. A number is the return of C-Steering minus that of the baseline when both are safe; C: only C-Steering is safe; B: only the baseline is safe; ·: neither. The third column is C-Steering's own normalised return.

Task	d	C-Steering	BC-Safe	BCQ-Lag	CPQ	COptiDICE	CDT	FISOR	O3SRL
BallCircle	10	$+0.76$	$+0.29$	C	$+0.12$	C	C	$+0.42$	$+0.15$
	40	$+0.89$	$+0.31$	$+0.18$	$+0.21$	C	$+0.05$	$+0.55$	$+0.17$
BallRun	10	$+0.08$	C	C	C	C	C	-0.05	-0.15
	40	$+0.04$	-0.31	C	-0.26	C	-0.49	-0.09	-0.21
CarCircle	10	$+0.76$	C	C	$+0.07$	C	C	$+0.37$	$+0.09$
	40	$+0.78$	$+0.23$	C	$+0.05$	C	-0.01	$+0.37$	$+0.12$
CarRun	10	unsafe	B	B	·	B	B	B	B
	40	$+0.98$	$+0.01$	$+0.04$	$+0.02$	$+0.05$	$+0.00$	$+0.28$	$+0.02$
DroneCircle	10	$+0.58$	$+0.09$	C	C	C	C	$+0.11$	$+0.10$
	40	$+0.64$	$+0.05$	C	$+0.86$	$+0.37$	-0.04	$+0.19$	$+0.07$
DroneRun	10	$+0.57$	$+0.15$	C	C	C	-0.01	C	C
	40	unsafe	B	·	·	·	·	B	B
AntCircle	10	$+0.56$	C	C	$+0.56$	C	C	$+0.32$	$+0.09$
	40	$+0.81$	$+0.39$	C	$+0.80$	C	C	$+0.60$	$+0.33$
AntRun	10	$+0.62$	C	C	$+0.60$	C	C	$+0.23$	$+0.33$
	40	$+0.69$	$+0.01$	C	$+0.67$	$+0.07$	-0.04	$+0.26$	$+0.46$
CarCircle1	10	$+0.56$	C	C	C	C	C	C	C
	40	$+0.55$	$+0.20$	C	C	C	C	C	C
CarCircle2	10	$+0.49$	C	C	C	C	C	C	C
	40	$+0.52$	C	C	C	C	C	C	C
PointCircle1	10	$+0.47$	$+0.17$	C	$+0.04$	C	-0.07	C	C
	40	$+0.48$	$+0.09$	C	$+0.05$	C	-0.09	C	C
PointCircle2	10	$+0.41$	C	C	C	C	C	C	C
	40	$+0.43$	C	C	C	C	C	C	C
CarGoal1	10	$+0.00$	-0.22	C	C	C	C	C	C
	40	unsafe	B	B	·	B	·	B	·
CarGoal2	10	$+0.07$	C	C	C	C	C	$+0.05$	C
	40	$+0.30$	$+0.16$	C	C	$+0.05$	C	$+0.26$	C
PointGoal1	10	$+0.64$	$+0.39$	C	$+0.12$	C	C	C	C
	40	$+0.78$	$+0.41$	$+0.06$	$+0.23$	C	$+0.09$	C	C
PointGoal2	10	unsafe	·	·	·	·	·	·	·
	40	$+0.65$	$+0.33$	C	C	C	C	$+0.37$	C
CarPush2	10	-0.03	C	C	C	C	C	C	C
	40	unsafe	B	·	·	·	·	B	·
PointPush1	10	$+0.10$	$+0.02$	C	C	C	C	C	C
	40	$+0.39$	$+0.24$	$+0.12$	$+0.15$	$+0.26$	$+0.11$	$+0.09$	$+0.08$
CarButton1	10	-0.01	C	C	C	C	C	$+0.06$	C
	40	-0.01	-0.04	C	C	C	C	$+0.03$	C
CarButton2	10	-0.01	C	C	C	C	C	-0.01	C
	40	-0.01	$+0.00$	C	C	C	C	$+0.00$	C
PointButton1	10	-0.16	C	C	C	C	C	C	C
	40	$+0.03$	-0.02	C	C	-0.07	C	-0.05	C
both safe			23	4	16	6	11	22	14
C-Steering higher / lower			18 / 4	4 / 0	15 / 1	5 / 1	3 / 7	17 / 4	12 / 2
only C-Steering safe (C)			14	33	21	31	26	15	23
only the baseline safe (B)			4	2	0	2	1	4	2
neither safe (·)			1	3	5	3	4	1	3