Policy Optimization in Hybrid Discrete-Continuous Action Spaces via Mixed Gradients

Matias Alvo
Graduate School of Business
Columbia University
malvo26@gsb.columbia.edu

Daniel Russo
Graduate School of Business
Columbia University
djr2174@gsb.columbia.edu

Yash Kanoria
Graduate School of Business
Columbia University
yk2577@gsb.columbia.edu

Abstract

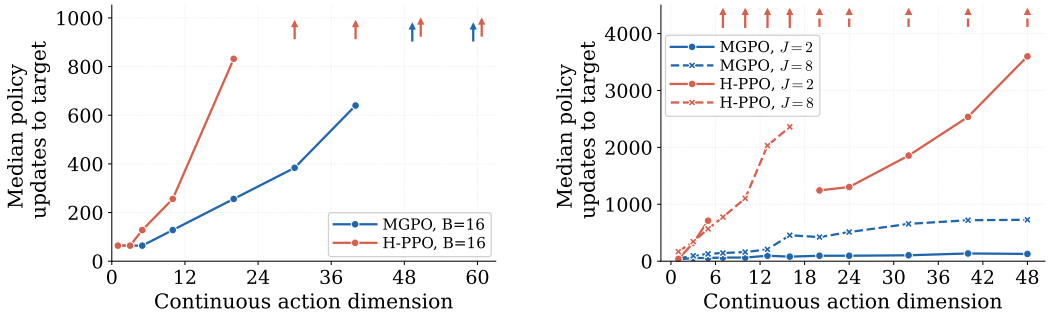
We study reinforcement learning in hybrid discrete–continuous action spaces, such as settings where the discrete component selects a regime (or index) and the continuous component optimizes within it — a structure common in robotics, control, and operations problems. Standard model-free policy gradient methods rely on score-function (SF) estimators and suffer from severe credit-assignment issues in high-dimensional settings, leading to poor gradient quality. On the other hand, differentiable simulation largely sidesteps these issues by backpropagating through a simulator, but the presence of discrete actions or non-smooth dynamics yields biased or uninformative gradients. To address this, we propose Mixed Gradient Policy Optimization (MGPO), which backpropagates through the simulator wherever smoothness permits, using a mixed gradient estimator that combines pathwise and SF gradients while maintaining unbiasedness. We also show how problems with action discontinuities can be reformulated in hybrid form, further broadening its applicability. Empirically, MGPO substantially outperforms Hybrid PPO on inventory control and switched linear-quadratic regulator problems, with performance gaps increasing as the continuous action dimension grows. Finally, we characterize the structure of the mixed gradient, showing that its cross term — which captures how continuous actions influence future discrete decisions — becomes negligible near a discrete best response, thereby enabling approximate decentralized updates of the continuous and discrete components and reducing variance near optimality.

1 Introduction

Reinforcement learning (RL) at scale faces a fundamental credit assignment problem. Methods such as REINFORCE [Williams, 1992] and Proximal Policy Optimization (PPO) [Schulman et al., 2017] estimate policy gradients by correlating perturbations in the policy's actions with the resulting cumulative reward. When a trajectory contains many decisions, each itself a high-dimensional action vector, this estimation problem becomes severely underdetermined: a single scalar return must be attributed back across thousands of action components whose effects are entangled with one another and with exogenous randomness. The signal-to-noise ratio of score-function (SF) gradient estimators degrades accordingly, and learning becomes slow or stalls altogether.

Recent work shows that this difficulty can be largely sidestepped when given a simulator that exposes additional structure beyond what a generic MDP provides. Two properties are central: exogeneity and

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



(a) Joint replenishment problem.

(b) Switched LQR, two mode counts J.

Figure 1: Median number of policy updates required for MGPO and H-PPO to reach 10% of the best observed run, as the continuous action dimension p increases, on the JRP (left) and switched LQR (right). We consider 10 independent trials per configuration. Vertical arrows represent configurations for which the median does not reach the target within the allotted budget.

smoothness. Under exogeneity, the underlying randomness — customer demand, weather, market movements — is independent of the agent's actions, so a recorded scenario can be replayed against any policy. Although the decision-maker (DM) still acts under partial information, the simulator has access to the full realization, enabling offline evaluation of arbitrary policies on the same trajectory. Under smoothness, small changes in actions induce small, computable changes in costs and state transitions, allowing gradients to be obtained by backpropagation through the simulator. Rather than relying on noisy outcome correlations, pathwise (PW) gradients compute how each action affects total cost via the chain rule, thereby largely mitigating the credit assignment problem. This paradigm has driven recent progress in robotics [Hu et al., 2019, Levine et al., 2016], inventory management [Alvo et al., 2023, Madeka et al., 2022], and queueing control [Che et al., 2024].

However, PW gradients are brittle to even mild departures from smoothness: discontinuities in the cost or transition functions introduce bias in the resulting gradients, and settings with purely discrete actions render PW gradients entirely uninformative. As soon as a problem fails to satisfy *all* the conditions for clean PW estimation, the standard practice is to fall back to fully model-free methods — often discarding the very structure that made differentiable simulation effective in the first place.

In practice, many problems satisfy **most** conditions allowing the use of PW gradients. For example, several problems involve a *hybrid* action space, in which some actions are discrete and others are continuous, and the model is smooth with respect to the continuous actions. One such example is the Switched Linear Quadratic Regulator (S-LQR) [Zhang et al., 2009], in which the DM selects a discrete *mode* that defines the transition and cost functions, and a continuous control input given the mode. Additionally, some problems have only a "few" discontinuities. For instance, in the joint replenishment problem (JRP) with large fixed ordering costs [Goyal and Satir, 1989], the cost function is discontinuous at zero but continuous elsewhere. In the settings outlined above, should we discard PW gradients entirely, or can we exploit them wherever structure permits?

Main contributions. In this work, we develop policy gradient algorithms for settings involving hybrid discrete-continuous action spaces. We propose a modeling framework and structured policy classes that enable an unbiased gradient estimator blending PW and SF gradients. Our design leverages PW gradients wherever possible, enabling sharper credit assignment where the problem structure allows. We call our framework *Mixed Gradient Policy Optimization* (MGPO). As shown in Figure 1, MGPO scales far better with the continuous action dimension than Hybrid PPO (H-PPO) [Fan et al., 2019], a hybrid-action extension of PPO, the go-to policy-based RL algorithm, with especially dramatic gaps in the S-LQR. Full results in Appendix D confirm that MGPO consistently outperforms H-PPO in high-dimensional settings. As we show in Section 4.2 (see Figure 2), this performance gap is driven by MGPO's *mixed* gradient estimator — which blends PW and SF terms — providing a drastically better gradient signal than pure SF-type gradients, which H-PPO relies on.

Our main contributions are as follows:

1. **Mixed Gradient Policy Optimization framework:** We propose a policy optimization framework for hybrid discrete–continuous action spaces. We show that when the environment is smooth in the state and continuous action, formulating the problem as an MDP with exogenous noise and pairing a stochastic discrete policy with a deterministic continuous one yields an unbiased *mixed* gradient estimator that combines PW and SF terms. To extend this framework to problems with action discontinuities, we show how a class of such problems can be expressed in hybrid form.
2. **Empirical evaluation:** We find that MGPO significantly outperforms H-PPO, with performance gains that increase with the continuous action dimension. We also find that the mixed gradient can yield much more accurate gradient estimation than pure SF estimators (see Figure 2), supporting the intuition that PW gradients improve credit assignment.
3. **Toward decentralized training:** The mixed gradient entangles the discrete and continuous components through a "cross term", necessitating centralized computation with privileged access to both. We prove this coupling vanishes near a discrete best response (Theorem 2), grounding approximate decentralized updates of the two components. The vanishing also motivates dropping the cross term as a variance-reduction technique (Section 5).

Overview of related work. Several works have tackled discrete–continuous action spaces using *model-free RL*. Xiong et al. [2018] introduced P-DQN, which combines DQN-style discrete action selection with deterministic policy-gradient updates for the continuous parameters. Fan et al. [2019] proposed H-PPO, with parallel discrete and continuous actors and a shared critic, trained with PPO. Huang et al. [2022a] proposed a related DQL–DDPG approach, using an actor to generate continuous controls for each candidate discrete action and a shared critic to evaluate the resulting hybrid actions. Li et al. [2021] proposed HyAR, which uses a conditional variational autoencoder to embed hybrid actions in a continuous latent space and optimizes the policy there with off-policy RL (TD3). These works treat the environment as a black box and therefore cannot exploit PW gradients, even when smoothness is available; we use H-PPO and HyAR as baselines in Section 4.

Our work builds on the literature of unbiased gradient estimation for stochastic objectives, a central topic in modern ML; see Mohamed et al. [2020] for an accessible introduction. Of particular importance is the stochastic computation graphs (SCG) framework of Schulman et al. [2015a], a general recipe for computing unbiased gradient estimators in computational graphs with both stochastic and deterministic operations. We build on this framework and specialize it to hybrid action spaces, deriving a "mixed" gradient estimator tailored to their structure. Levy and Ermon [2018] derive mixed gradient estimators for problems with discrete actions, but do so by introducing a continuous relaxation of the dynamics and optimizing the resulting surrogate objective. We instead exploit the structure of hybrid action spaces: smoothness holds conditional on the discrete action, which suffices to obtain an unbiased estimator for the original objective without relaxing the dynamics.

Our work also broadens the applicability of differentiable simulation [Hu et al., 2019, Freeman et al., 2021, Alvo et al., 2023, Madeka et al., 2022], which can substantially outperform model-free methods in high-dimensional continuous control [Xu et al., 2022]. Prior work has extended differentiable simulation to discrete decisions using surrogate gradients such as the straight-through (ST) estimator [Bengio et al., 2013] (see, e.g., Che et al. [2024] for an application to queueing networks) and the Gumbel-Softmax trick [Maddison et al., 2016, Jang et al., 2016]. However, these relaxations yield biased gradients: for ST, the bias need not vanish as the relaxation is tightened (see Section 3 for an example in inventory control), while fully soft relaxations reduce bias only at the cost of high gradient variance [Jang et al., 2016]. In contrast, we exploit problem structure in hybrid discrete–continuous action spaces, leveraging PW gradients wherever differentiability is available in a manner that preserves unbiasedness. Additionally, a recent line of work improves the conditioning of PW gradients over long horizons and contact-rich dynamics by truncating backpropagation and bootstrapping with a learned critic [Xu et al., 2022, Xing et al., 2025]. Our work is complementary: we make PW gradients available when part of the action is discrete, and we expect such techniques to apply to the resulting PW term.

Limited scope. Our work follows a line of research that assumes access at training time to an exogenous simulator, enabling replay of alternative policies under the same realization of uncertainty (Sec. 2). Our methods do not apply to fully black-box simulators, or to dynamics that cannot be decomposed into smooth pieces (see Remark 1), where model-free methods remain the appropriate choice. Within the exogenous-simulator setting, our central question is whether the benefits of differentiable simulation can be preserved in hybrid discrete–continuous action spaces. To answer this

question, we compare MGPO with two popular model-free algorithms tailored to hybrid action spaces, H-PPO and HyAR. Finally, we focus on benchmarks (JRP and S-LQR) that allow systematic variation of the continuous action dimension, isolating its effect on gradient quality and optimization behavior. A natural next step is to apply this framework to large-scale problems in multi-echelon inventory control, queueing control, and robotic control, where differentiable simulators are increasingly available but discrete decisions have limited the reach of PW methods.

2 Problem Description

We consider a setting recently dubbed an *exogenous MDP* — less a property of the underlying control problem than a description of the *simulator available at training time*. Classical dynamic programming assumes the transition model is analytically tractable, while standard RL treats the environment as a black box with no ability to reuse randomness across policies. An exogenous MDP sits between these extremes: the simulator evaluates policies on a fixed *scenario* (a realization of the underlying exogenous randomness) using known dynamics, enabling direct policy comparison on the same trajectory and sharper credit assignment. *Crucially, the DM does not observe the scenario at decision time — the simulator's privileged access is a training-time artifact, not a relaxation of the information structure.* This structure arises naturally in robotics (scenarios fix physical parameters; the agent observes state and acts), logistics (known flow or service dynamics), and inventory control (scenarios are demand realizations). A summary of the notation used throughout the paper is provided in Table 1 (Appendix A.1). We formalize this setting as follows.

2.1 Markov decision processes with exogenous randomness

There are two common (and equivalent) mathematical abstractions of an MDP: one specified by a transition kernel $P(\cdot \mid s, a)$, and one written in "disturbance" form as a deterministic transition map f driven by exogenous noise ξ_t [Bertsekas, 2012]. We adopt the latter formalism, since it streamlines the development of our gradient estimators and algorithms.

In each period t, a DM observes state $s_t \in \mathcal{S}$, chooses action $a_t \in \mathcal{A}(s_t)$, and the system evolves as $s_{t+1} = f(s_t, a_t, \xi_t)$ with cost $c(s_t, a_t, \xi_t)$, where $\xi = (\xi_0, \ldots, \xi_{T-1})$ is a sequence of exogenous random terms with joint law $\mathbb{P}_\xi$. We denote an MDP by $\mathcal{M} = (\mathcal{S}, \mathcal{A}, c, f, \mathbb{P}_\xi)$ and define the expected cost $J(\pi)$ for initial state s_0, which the DM seeks to minimize over $\pi \in \Pi$:

$$J(\pi) := \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t \, c(s_t, \pi_t(s_t), \xi_t) \,\middle|\, s_0 \right].$$

We next impose a substantive assumption on the training-time simulator. Following *hindsight RL* [Alvo et al., 2023, Xie et al., 2024] and *exogenous MDPs* [Sinclair et al., 2023], we assume that the simulator has access to the transition and cost functions (f, c) and observes the realized exogenous disturbances. Therefore, it can evaluate any policy counterfactually on the same fixed scenario $\bar{\xi}$.

2.2 Hybrid discrete-continuous action MDPs

Our methods apply to MDPs with hybrid discrete–continuous action spaces, which we call *hybrid MDPs*. The state s is a continuous vector and the action is a pair $a = (x, b)$, where x is a discrete action selected from a finite set $\mathcal{X}$ and b is a continuous action; the set of feasible continuous actions $\mathcal{B}^x$ may depend on x. In addition to the simulator access assumed in Section 2.1, our primary assumption is that the transition and cost functions are smooth with respect to s and b, but not necessarily x: that is, $f(s, (x, b), \xi)$ and $c(s, (x, b), \xi)$ are smooth in (s, b) for any fixed x and ξ. Formal conditions are provided in Appendix A.2.

We illustrate the framework with two examples that serve as our main benchmarks.

Example 1 (Joint replenishment problem)**.** A retailer manages p products over T periods. In each period, it observes current inventory levels and outstanding (replenishment) orders, which arrive after a known lead time. It must decide whether to place an order (the discrete action), incurring a fixed cost K, and, if so, how many units of each product to order (the continuous action). The goal is to minimize the expected sum of fixed ordering costs and the costs of overstocking and understocking. A scenario consists of the initial state and the exogenous demand sequence. We provide two alternative

formulations in Appendix A.5: one as a hybrid MDP, and one with a purely continuous action, in which the fixed cost makes the cost function discontinuous (see Remark 1).

Example 2 (Switched LQR)**.** This generalizes the classical LQR problem [Kalman, 1960]: a controller chooses both a discrete operating mode, which determines the dynamics and cost matrices, and a continuous input to keep the state near zero while limiting control effort. At each period $t \in \{0, \dots, T-1\}$, the agent observes state $s_t \in \mathbb{R}^p$, selects a discrete mode $x_t \in \{1, \dots, J\}$ and a continuous control $b_t \in \mathbb{R}^p$. Each mode j has dynamics matrices (A_j, B_j) and positive-definite cost matrices (Q_j, R_j), and the system evolves as

$$s_{t+1} = A_{x_t} s_t + B_{x_t} b_t + W_t, \qquad c(s_t, a_t) = s_t^\top Q_{x_t} s_t + b_t^\top R_{x_t} b_t,$$

where W_t is exogenous process noise. The exogenous scenario consists of s_0 and $W_{0:T-1}$.

Remark 1 (Piecewise-smooth MDPs (PS-MDPs)). The hybrid MDP framework also encompasses problems in which the cost or transition functions are discontinuous in the action but smooth within finitely many regions of the action space, which we refer to as PS-MDPs. Such problems can be recast as hybrid MDPs by letting the discrete action select the smooth region and the continuous action specify the control within it. A formal definition and equivalence result are provided in Appendix A.3.

3 Policy approximation and gradient computation

Policy representation. A policy in a hybrid MDP maps states to a joint distribution over discrete and continuous actions. Although these components generally do not decouple — the continuous control should depend on the selected mode, as in the S-LQR — we find it useful to factor the policy as $\pi = (\pi^{\mathcal{X}}, \pi^{\mathcal{B}})$, where $\pi^{\mathcal{X}}$ outputs a distribution over discrete actions and $\pi^{\mathcal{B}}$ deterministically outputs one action *candidate* per discrete value $j \in \mathcal{X}$. The DM samples $x_t \sim \pi^{\mathcal{X}}(s_t)$ and applies the candidate $\pi^{\mathcal{B}}(s_t)[x_t]$ (see Figure 4 in Appendix C.1). We consider a parameterized hybrid policy π_θ with $\theta = (\phi, \kappa)$, where $\pi^{\mathcal{X}}_\phi$ and $\pi^{\mathcal{B}}_\kappa$ are parameterized independently. Parameter sharing is possible but would obscure the interplay between the two components central to our analysis (Section 5).

Pathwise gradient estimation. Suppose the discrete policy were fixed at $x_t = 1$ for all t. Then, for a fixed scenario ξ, the system evolves as the smooth deterministic dynamical system $s_{t+1} = f(s_t, (1, \pi^{\mathcal{B}}_\kappa(s_t)[1]), \xi_t)$, and the state $s_t(\kappa)$ is a smooth function of κ computed by iterating f. The total cost $\sum_{t=0}^{T-1} c(s_t(\kappa), (1, \pi^{\mathcal{B}}_\kappa(s_t(\kappa))[1]), \xi_t)$ is also smooth in κ. Its gradient with respect to κ — obtained by direct backpropagation through the simulator — is called the *pathwise (PW) gradient*. Unlike SF, PW gradients isolate each action coordinate's effect on cost, sharpening credit assignment.

Deriving the mixed policy gradient estimator. Towards clean mathematical formalizations including Theorem 1 below, we adopt a discounted infinite-horizon formulation with $\gamma \in (0, 1)$ and stationary policies (our experiments will operate on finite-horizon trajectories of length T). Standard regularity of dynamics and policies—smoothness of f, c in (s, b); differentiability and bounded log-gradients of the policy components; bounded costs—is formalized as Assumptions 2 and 3 in Appendices A.2 and B.1, respectively.

When the discrete component is non-trivial and stochastic, the scenario ξ alone no longer determines a smooth trajectory: the sampled $x_{0:\infty}$ also varies. The key insight is that the within-trajectory smoothness established above still holds conditioned on any realization of $(x_{0:\infty}, \xi)$. For the formal statement, fix an arbitrary scenario ξ and discrete sequence $x_{0:\infty}$. Conditional on this realization, the state $s_t(\kappa)$ and continuous action $b_t(\kappa)$ are deterministic smooth functions of κ, just as in the purely continuous case above. The theorem statement suppresses the dependence on $(\xi, x_{0:\infty})$, focusing on the dependence of the realized trajectory on κ. We write $\mathbb{E}_\theta$ for expectation under the joint law of $(\xi_{0:\infty}, x_{0:\infty})$ induced by π_θ, set $J(\theta) := \mathbb{E}_\theta\big[\sum_{t=0}^{\infty} \gamma^t c(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t) \mid s_0\big]$, and let Q_{π_θ} denote its action-value function.

Theorem 1 (Mixed policy gradient estimator)**.** *Under Assumptions 2 and 3, the objective $J(\theta)$ is differentiable in $\theta = (\phi, \kappa)$ and its gradients are given by:*

$$\nabla_\phi J(\theta) = \mathbb{E}_\theta \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\phi \log \pi_\phi^{\mathcal{X}} \left(x_t \mid s_t(\kappa) \right) Q_{\pi_\theta}(s_t(\kappa), a_t) \right], \tag{1}$$

$$\nabla_\kappa J(\theta) = \underbrace{\mathbb{E}_\theta \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\kappa c\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big) \right]}_{\textit{PW gradient term}} + \underbrace{\mathbb{E}_\theta \left[\sum_{t=0}^{\infty} \gamma^t \nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa)) \, Q_{\pi_\theta}(s_t(\kappa), a_t) \right]}_{\textit{Cross term}} \tag{2}$$

We provide a full proof of Theorem 1 in Appendix B.2. Although the proof follows directly from the SCG framework [Schulman et al., 2015a] (see Weber et al. [2019] for its extension to critics), the mixed gradient structure arises only from our modeling choices: formulating the problem as a hybrid MDP with exogenous noise, smooth in the state and continuous action (see Appendix A.2 for formal conditions), and pairing a stochastic discrete policy with a deterministic continuous one.[1] These choices can be viewed as a *partial* application of the reparameterization trick [Kingma and Welling, 2013, Rezende et al., 2014]: conditioning on the exogenous noise makes the trajectory differentiable in the continuous action, even in the presence of discrete actions. In contrast, applying the same framework to a stochastic policy whose continuous component is not reparameterized, as in standard model-free RL, yields the ubiquitous SF estimator (see Appendix B.4).

The gradient with respect to ϕ is the standard SF estimator. The *pathwise term* captures the direct effect of κ on costs and the state trajectory, holding discrete actions fixed. The *cross term* captures how changes in κ alter $s_t(\kappa)$, shifting the probabilities the discrete policy assigns to each action, weighted by the value Q_{π_θ} of the resulting actions. Section 5 studies the effect of dropping this term.

Building on Theorem 1, Listing 1 shows how to construct a loss whose gradient yields the mixed estimator; the full training procedure is given in Algorithm 1 (Appendix C.2). The sampled discrete actions x_t are held fixed, so no derivative is taken through sampling or through the index selecting b_t: differentiating $\log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))$ with respect to κ through the state trajectory and weighting it by an advantage estimate (subtracting a state-dependent baseline from Q_{π_θ} leaves the expectation unchanged) yields the cross term, while differentiating the realized costs yields the PW term. In the listing, `stop_grad` denotes PyTorch's `detach`; the flag `drop_cross` detaches the state only inside the discrete log-probability, removing the cross term while leaving the PW gradient for κ and the SF gradient for ϕ unchanged. We use **MGPOFull** and **MGPONoCross** to denote MGPO with and without the cross term, respectively.

Pitfalls of discrete-action smoothing. PW gradients can be applied in hybrid settings by smoothing the discrete action. However, such smoothing requires the simulator to execute fractional actions, which are typically infeasible, and trades bias at high temperatures for high gradient variance at low ones [Jang et al., 2016]. The standard alternative is the straight-through (ST) estimator [Bengio et al., 2013], which executes the hard discrete action in the forward pass but differentiates as if it were continuous in the backward pass. Even as the temperature vanishes, ST-type estimators replace the true difference in cost between discrete actions with the relaxed cost's derivatives at the two endpoints [Shekhovtsov, 2023]. In inventory control with fixed ordering costs, whether to place an order should depend on the fixed cost and on the value of *every* unit the order brings in. Because ST's forward pass commits to either no order or the full order, its backward pass only evaluates the marginal value of inventory at these two extremes, i.e., for the first and last units of the order. Unless this marginal value varies linearly across the order, ST remains biased and can select an incorrect reorder point.

4 Numerical experiments

Problem settings. We empirically evaluate MGPO on the two problem classes introduced in Section 2.2, demonstrating performance gains over H-PPO as the continuous action dimension increases. We consider the JRP (Example 1) [Goyal and Satir, 1989], a classical problem in operations management where fixed costs create interdependencies across products that make scaling fundamentally

[1] Reparameterizable stochastic continuous policies are covered by absorbing their randomness into the exogenous scenario.

Listing 1: MGPO gradient estimation.

```
def MGPO_loss(policy, value_fn, xi, s0, gamma, drop_cross=False):
    s, loss = s0, 0
    for t in range(len(xi)):
        # If drop_cross=True, detach s only inside the discrete score:
        # this removes the kappa cross term but keeps the phi gradient.
        s_score = stop_grad(s) if drop_cross else s

        logits = policy.discrete(s_score)
        x = Categorical(logits).sample() # discrete action, treated as fixed
        logp = log_prob(logits, x) # autograd flows through logits, not x

        b = policy.continuous(s)[x] # autograd flows through selected candidate
        cost = c(s, x, b, xi[t]) # smooth for fixed x
        s_next = f(s, x, b, xi[t]) # smooth for fixed x

        # Illustrating with a TD-style advantage; this weight is not differentiated.
        advantage = stop_grad(cost + gamma * value_fn(s_next) - value_fn(s))
        loss += gamma**t * cost # pathwise term for kappa
        loss += gamma**t * advantage * logp # phi score; kappa cross unless detached

        s = s_next
    return loss
loss = MGPO_loss(policy, value_fn, xi, s0, gamma, drop_cross=False)
loss.backward() # phi: score term; kappa: pathwise + cross term
# use drop_cross=True to remove only the kappa cross term
```

difficult [Ata et al., 2025], and where PPO has been shown to underperform simple heuristics [Vanvuchelen et al., 2023]. We also consider the S-LQR (Example 2) [Zhang et al., 2009], which extends the classical LQR [Kalman, 1960] — a canonical control benchmark — to settings with discrete operating modes; finding the optimal mode sequence is NP-hard [Zhang et al., 2011].

Policy optimization. Throughout our experiments, we consider **MGPOFull**, and refer to it simply as MGPO. As our main baseline, we use H-PPO, built on the CleanRL PPO implementation [Huang et al., 2022b]. H-PPO shares MGPO's policy architecture, critic, and discrete-action update, differing only in its continuous head: stochastic and trained with SF gradients, rather than deterministic and trained with PW gradients. We also evaluated HyAR in both settings, but it performed substantially worse than H-PPO overall; we report its results in Appendix D.5.

Our implementation of MGPO uses standard PPO to optimize the discrete policy component. When computing $\nabla_\kappa J$, we substitute the GAE-based advantage estimates produced by PPO into the SF loss term of Listing 1. We train across batches of scenarios, updating the continuous parameters with one full-batch policy update using the mixed estimator, while the discrete parameters are updated using 5 PPO epochs with 4 minibatches per epoch. Full hyperparameters and implementation details are provided in Appendix C.2. Code is available at `https://github.com/MatiasAlvo/hybrid-rl`.

4.1 Finding 1: MGPO converges faster as the continuous action dimension grows

We compare MGPO and H-PPO on the JRP (Example 1) and the S-LQR (Example 2), varying the continuous action dimension p across 10 independent trials per configuration. Our primary metric is the median number of policy updates required to reach within 10% of the best validation loss achieved by any algorithm at each value of p; test loss across configurations is reported in Appendix D.

JRP. Figure 1a reports the median number of policy updates to reach a 10% target gap, using batch size $B = 16$ and trajectories of 100 periods, for p ranging from 1 to 60. The two algorithms perform comparably at small dimensions, but their scaling behavior diverges as p grows: H-PPO's required policy updates rise steeply with p, while MGPO's remains substantially flatter, though MGPO with $B = 16$ also fails to converge within the training budget for $p \geq 50$. Figure 5 in Appendix D.1 extends this comparison to batch sizes $B \in \{16, 64\}$ and a range of target gaps, revealing that for $p \geq 40$, MGPO with $B = 16$ converges faster than H-PPO with $B = 64$ at easier target gaps. We hypothesize this stems from the superior quality of the mixed gradient estimator over the SF-type

gradients used by H-PPO, which we analyze directly in Section 4.2. Tables 2 and 3 further corroborate these findings via test loss: MGPO with $B = 16$ achieves lower median test loss than H-PPO with $B = 64$ for $p \geq 50$. Finally, Figure 12 in Appendix D.4 shows that hyperparameter tuning does not close the gap, suggesting the degradation is a fundamental limitation of SF-based optimization.

S-LQR. Figure 1b reports the same metric for p ranging from 1 to 48, mode counts $J \in \{2, 8\}$, and batch size 128. The performance gap is drastic. For $J = 8$, H-PPO degrades sharply once p exceeds roughly 10, with the median failing to converge for $p \geq 20$. For $J = 2$, H-PPO gets stuck at bad local optima for intermediate values ($7 \leq p \leq 16$); for larger p it converges again, but needs up to 3600 policy updates at $p = 48$. Figure 6 in Appendix D.2 further shows that H-PPO with $J = 8$ fails to reach even a 30% target gap for $p \geq 24$. MGPO, by contrast, reaches a 5% target across all configurations including $p = 48$, within roughly 1000 policy updates.

In Appendix D.2.1, we validate MGPO against a Riccati-based baseline. For $J = 1$ (standard LQR), where the Riccati solution is optimal, MGPO recovers performance within 0.2% at $p = 48$ (Figure 8a). For $J = 4$, MGPO improves upon the best single-mode Riccati baseline by 19% to 38% (Figure 8b), demonstrating that it learns to exploit mode switching. Full results are in Appendix D.2.

4.2 Finding 2: MGPO yields superior gradient estimates as the action dimension grows

We now investigate *why* MGPO converges faster than H-PPO. The key claim is that backpropagating through the simulator sharpens credit assignment, increasingly so as the continuous action dimension grows. We test this by comparing the mixed and SF estimators of $\nabla_\kappa J(\theta)$ on the JRP.

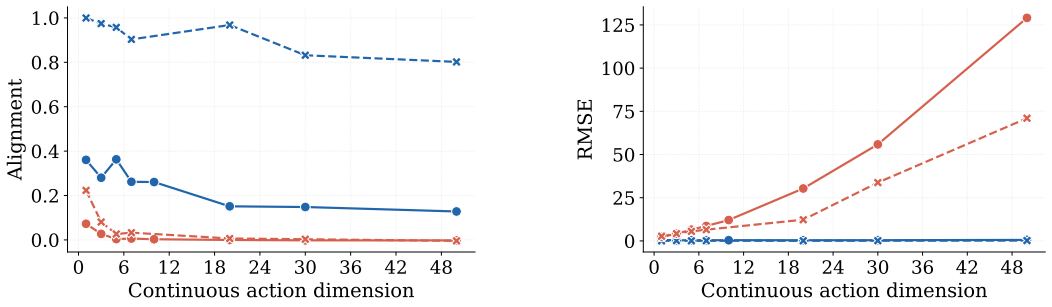
Experimental setup. From Eq. (2), the mixed gradient combines a PW term with a cross term involving gradients of the discrete policy's log-likelihoods. As $\pi_\phi^{\mathcal{X}}$ becomes more peaked — as it does when the discrete policy approaches a best response — these log-likelihood gradients grow large near the boundaries between discrete actions, potentially inflating the cross term's variance. The signal quality of the mixed estimator may therefore vary substantially with the quality of the discrete component. We evaluate this along a natural training path in which the continuous policy is held fixed while the discrete policy improves, isolating the effect of discrete-policy quality from that of the continuous-action dimension p. Concretely, we train a policy to near-optimality, then make the continuous mapping stochastic by adding Gaussian noise so that the SF estimator for κ is well-defined, and apply a fixed offset to the continuous outputs so that $\nabla_\kappa J$ is nonzero. We then re-initialize the discrete network and train it to convergence, saving checkpoints bucketed by performance gap, and report results stratified by this gap. Full details are provided in Appendix D.3.

Estimators. To apply the *mixed* estimator, we use the reparameterization trick: under $b_t = \mu_\kappa(s_t) + \sigma\varepsilon_t$ with $\varepsilon_t \sim \mathcal{N}(0, I)$, the policy noise is treated as additional exogenous randomness absorbed into the scenario. Conditioned on $(x_{0:T-1}, \xi_{0:T-1}, \varepsilon_{0:T-1})$, the trajectory is deterministic and differentiable in κ, so the mixed-gradient derivation applies verbatim. For SF, we consider a learned critic baseline to reduce variance; we omit PPO-style modifications, which bias the estimate.

Findings. Figure 2 reports two metrics, each estimated from pairs of independent batch-level gradient estimates using batches of 512 trajectories. **Alignment** is the expected cosine similarity between two independent estimates, measuring directional consistency. **RMSE** is the total standard deviation of the estimator across all coordinates, which can be estimated from pairs of independent estimates without access to the true gradient (see Appendix D.3 for details).

SF exhibits alignment above 0.05 only for small problem sizes ($p = 1, 3$), and quickly collapses to near zero as the dimension grows. This behavior is consistent across performance gaps and indicates that SF provides little directional signal in high-dimensional settings. Similarly, its RMSE increases rapidly with p, which contributes to the low alignment values. The deterioration in both metrics closely mirrors H-PPO's performance degradation observed in Figure 1, suggesting that the gradients used by H-PPO suffer from poor signal quality in high-dimensional settings.

In contrast, for performance gaps of 30–35%, the mixed estimator maintains alignment above 0.8 across all values of p, along with consistently low RMSE, indicating a stable gradient direction even in high-dimensional regimes. Both metrics degrade for smaller gaps (5–10%), but remain significantly better than those of SF. We hypothesize that this degradation is driven by increased variance in the cross term as the discrete policy approaches a best response, which we analyze in Appendix D.3.6.



(a) Alignment **(higher is better)**.

(b) RMSE **(lower is better)**.

Figure 2: Alignment and RMSE of batch-level gradient estimates for the mixed and SF estimators for varying continuous action dimensions. Blue and red lines correspond to the mixed and SF estimators. Solid and dashed lines denote performance gap ranges of 5–10% and 30–35%, respectively.

5 The cross term vanishes for near-best response discrete policy components

In many hybrid MDPs, the two components of the action play distinct roles: the discrete component selects a regime — a controller, an operating mode, or a strategic choice — while the continuous component optimizes within it. In practice, distinct teams are often responsible for building a controller for each component. This raises a natural question: *can one have a decomposed training procedure in which each component is optimized separately using only behavioral access to the other*, that is, using observed states and actions without privileged access to the other component's policy?

The obstacle to such decomposition lies in the cross term of the mixed gradient. While the discrete component can be optimized using standard SF estimators based only on observed actions and costs, the cross term involves $\nabla_\kappa \log \pi_\phi^{\mathcal{X}}$, capturing how perturbations of the continuous action influence discrete decisions along the generated trajectories. Computing this term requires access to the internal structure of the discrete policy, not just its realized actions. Fortunately, as we show below, the cross term becomes small near a discrete best response, enabling approximate decentralization.

We now provide a second motivation for dropping the cross term. The cross term's estimation variance grows as the discrete policy approaches a best response (Section 4.2, Appendix D.3.6). Combined with the vanishing-expectation result we prove below, one is estimating a vanishing signal with growing noise, and can improve gradient quality near optimality by dropping the cross term.

Before we state our result, we give some necessary definitions and assumptions. For fixed continuous policy parameters κ, define the (discrete policy) *best-response* $\pi^{\mathcal{X}}(\kappa) \in \arg\min_{\pi^{\mathcal{X}}} J(\pi^{\mathcal{X}}, \kappa)$, and the corresponding policy $\pi_\kappa^* = (\pi^{\mathcal{X}}(\kappa), \pi_\kappa^{\mathcal{B}})$. We say that $\pi_\phi^{\mathcal{X}}$ is an *ϵ-best response* to κ if $\mathbb{E}_{s_0\sim\rho}[V_{\pi_\theta}(s_0) - V_{\pi_\kappa^*}(s_0)] \le \epsilon$ for a given initial state distribution ρ.

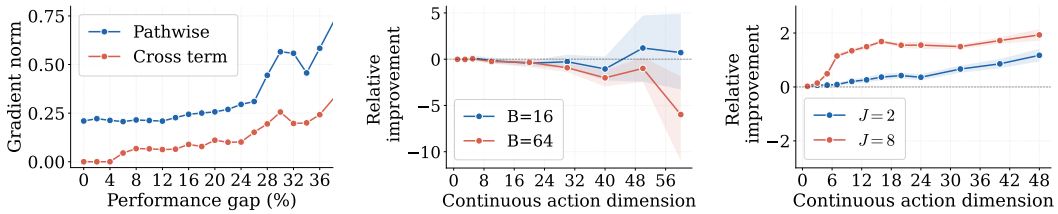
Assumption 1 (Regularity conditions). There exist constants $C, L_\pi, L_s < \infty$ such that, for all $\kappa \in \mathcal{K}$:

1. **Statewise best response.** There exists best-response $\pi^{\mathcal{X}}(\kappa)$ such that for all $s \in \mathcal{S}$, $V_{\pi_\kappa^*}(s) = \min_{\pi^{\mathcal{X}}} V_{(\pi^{\mathcal{X}}, \pi_\kappa^{\mathcal{B}})}(s)$.
2. **Uniform near-best response.** If $\pi_\phi^{\mathcal{X}}$ is any ϵ-best response to κ, then for any initial state distribution ρ', $\mathbb{E}_{s_0\sim\rho'}[V_{\pi_\theta}(s_0) - V_{\pi_\kappa^*}(s_0)] \le C\epsilon$.
3. **Policy smoothness and state sensitivity.** $\|\nabla_\kappa s_t(\kappa)\| \le L_s$ for all $t \ge 1$, $\xi_{0:t-1}$ and $x_{0:t-1} \in \mathcal{X}^t$, and $\sup_{\phi,s,x} \|\nabla_s \log \pi_\phi^{\mathcal{X}}(x \mid s)\| \le L_\pi$.

We now state our result.

Theorem 2 (Cross term vanishes for near-best responses)**.** *Under Assumption 1, for any κ and any ϵ-best response $\pi_\phi^{\mathcal{X}}$,*

$$\left\| \mathbb{E}_\theta \left[\sum_{t\ge 0} \gamma^t \nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))\; Q_{\pi_\theta}(s_t(\kappa), a_t) \right] \right\| \le \epsilon \times L_\pi L_s \left(1 + \frac{C\gamma}{1-\gamma}\right).$$



(a) Norm vs. performance gap. (b) Relative improvement (JRP). (c) Relative improvement (S-LQR).

Figure 3: (a) Gradient norms of the PW and cross terms for varying performance gaps in the JRP setting. (b) Cross term ablation in the JRP across batch sizes B. (c) Cross term ablation in the S-LQR across mode counts J. Improvement is measured as the percentage reduction for **MGPONoCross** relative to **MGPOFull**; positive values indicate better performance with **MGPONoCross**.

The proof in Appendix B.3 shows that the magnitude of the cross term is controlled by the suboptimality of the discrete policy relative to a best response, and by the sensitivity of the score term to the state and of the state to κ. These boundedness assumptions are reasonable under smooth stochastic policies (e.g., with entropy regularization), which prevent sharp decision boundaries. Analyzing the behavior near deterministic limits is more subtle and we leave it to future work.

Our result may be viewed as an envelope-style property for hybrid control problems in dynamic settings [Milgrom and Segal, 2002, Danskin, 1966], suggesting that the coupling between discrete and continuous decisions weakens as the discrete policy approaches a best response.

To illustrate Theorem 2 empirically, Figure 3a reports the gradient norms of the PW and cross terms in the setting of Section 4.2 with $p = 20$. As the discrete policy approaches a best response, the cross term norm decreases to zero even though the overall policy is not yet optimal, as indicated by the nonzero PW norm. This suggests that the continuous component can be updated using only the PW gradient, incurring only a small bias, and supports the view that near a discrete best response the problem admits an approximately decentralized optimization.

To assess the impact of removing the cross term, Figures 3b and 3c report the relative change in performance when using **MGPONoCross** in place of **MGPOFull** on the JRP and S-LQR experiments of Section 4.1, respectively. In the JRP setting, removing the cross term typically yields a slight performance degradation of at most 2%; the one exception is the combination $p = 60$, $B = 64$, where the degradation reaches 6%. These drops suggest that the discrete policy is not yet near a best response in this setting, so the cross term carries meaningful gradient signal. In the S-LQR problem, by contrast, removing the cross term is consistently beneficial, with gains that generally grow with p. We analyze these effects in Appendix D.3.6, where we show that dropping the cross term may introduce bias but tends to yield more stable updates in practice.

Combined with the empirical observation in Appendix D.3.6 that the cross term's *variance* grows as the discrete policy approaches a best response (as previewed at the start of this section), this implies that near optimality we are estimating a vanishing quantity with high-variance noise — providing a clear rationale for **MGPONoCross** to drop the cross term even in fully centralized settings.

6 Conclusion

Model-free policy-gradient methods such as REINFORCE and PPO struggle in large action dimensions, driven by the poor quality of the SF-type gradients they rely on. In this work, we propose MGPO, which leverages mixed gradients for hybrid discrete-continuous action MDPs. Empirically, MGPO overcomes the shortcomings of SF-based gradients in high-dimensional continuous action spaces, representing a significant step towards the applicability of policy-gradient methods in problems of realistic size and complexity. We establish that the cross term of the mixed gradient vanishes as the discrete policy approaches a best response, a property which may have implications both for the design of semi-decentralized RL algorithms and for improving gradient quality.

Acknowledgments and Disclosure of Funding

This work received no specific third-party funding. The authors declare no competing interests.

References

M. Alvo, D. Russo, Y. Kanoria, and M. Lee. Deep reinforcement learning for inventory networks: Toward reliable policy optimization. *arXiv preprint arXiv:2306.11246*, 2023.

B. Ata, W. van Eekelen, and Y. Zhong. A computational method for solving the stochastic joint replenishment problem in high dimensions. *arXiv preprint arXiv:2511.11830*, 2025.

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

D. Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 1. Athena scientific, 2012.

E. Che, J. Dong, and H. Namkoong. Differentiable discrete event simulation for queuing network control. *arXiv preprint arXiv:2409.03740*, 2024.

J. M. Danskin. The theory of max-min, with applications. *SIAM Journal on Applied Mathematics*, 14 (4):641–664, 1966.

Z. Fan, R. Su, W. Zhang, and Y. Yu. Hybrid actor-critic reinforcement learning in parameterized action space. *arXiv preprint arXiv:1903.01344*, 2019.

C. D. Freeman, E. Frey, A. Raichuk, S. Girgin, I. Mordatch, and O. Bachem. Brax–a differentiable physics engine for large scale rigid body simulation. *arXiv preprint arXiv:2106.13281*, 2021.

S. K. Goyal and A. T. Satir. Joint replenishment inventory control: deterministic and stochastic models. *European journal of operational research*, 38(1):2–13, 1989.

Y. Hu, L. Anderson, T.-M. Li, Q. Sun, N. Carr, J. Ragan-Kelley, and F. Durand. Difftaichi: Differentiable programming for physical simulation. *arXiv preprint arXiv:1910.00935*, 2019.

C. Huang, H. Zhang, L. Wang, X. Luo, and Y. Song. Mixed deep reinforcement learning considering discrete-continuous hybrid action space for smart home energy management. *Journal of Modern Power Systems and Clean Energy*, 10(3):743–754, 2022a.

S. Huang, R. F. J. Dossa, C. Ye, J. Braga, D. Chakraborty, K. Mehta, and J. G. Araújo. Cleanrl: High-quality single-file implementations of deep reinforcement learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022b.

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

R. E. Kalman. Contributions to the theory of optimal control. *Bol. soc. mat. mexicana*, 5(2):102–119, 1960.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

D. Levy and S. Ermon. Deterministic policy optimization by combining pathwise and score function estimators for discrete action spaces. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

B. Li, H. Tang, Y. Zheng, J. Hao, P. Li, Z. Wang, Z. Meng, and L. Wang. Hyar: Addressing discrete-continuous action reinforcement learning via hybrid action representation. *arXiv preprint arXiv:2109.05490*, 2021.

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

D. Madeka, K. Torkkola, C. Eisenach, A. Luo, D. P. Foster, and S. M. Kakade. Deep inventory management. *arXiv preprint arXiv:2210.03137*, 2022.

P. Milgrom and I. Segal. Envelope theorems for arbitrary choice sets. *Econometrica*, 70(2):583–601, 2002.

S. Mohamed, M. Rosca, M. Figurnov, and A. Mnih. Monte carlo gradient estimation in machine learning. *The Journal of Machine Learning Research*, 21(1):5183–5244, 2020.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pages 1278–1286. PMLR, 2014.

H. Scarf. The optimality of (S, s) policies in the dynamic inventory problem. In K. J. Arrow, S. Karlin, and P. Suppes, editors, *Mathematical Methods in the Social Sciences*, pages 196–202. Stanford University Press, Stanford, CA, 1960.

J. Schulman, N. Heess, T. Weber, and P. Abbeel. Gradient estimation using stochastic computation graphs. *Advances in neural information processing systems*, 28, 2015a.

J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

A. Shekhovtsov. Cold analysis of rao-blackwellized straight-through gumbel-softmax gradient estimator. In *International Conference on Machine Learning*, pages 30931–30955. PMLR, 2023.

S. R. Sinclair, F. V. Frujeri, C.-A. Cheng, L. Marshall, H. D. O. Barbalho, J. Li, J. Neville, I. Menache, and A. Swaminathan. Hindsight learning for mdps with exogenous inputs. In *International Conference on Machine Learning*, pages 31877–31914. PMLR, 2023.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

N. Vanvuchelen, B. De Moor, and R. Boute. The use of continuous action representations to scale deep reinforcement learning for inventory control. 2023.

T. Weber, N. Heess, L. Buesing, and D. Silver. Credit assignment techniques in stochastic computation graphs. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2650–2660. PMLR, 2019.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Reinforcement learning*, pages 5–32, 1992.

Y. Xie, W. Ma, and L. Xin. Vc theory for inventory policies. *arXiv preprint arXiv:2404.11509*, 2024.

E. Xing, V. Luk, and J. Oh. Stabilizing reinforcement learning in differentiable multiphysics simulation. In *International Conference on Learning Representations*, volume 2025, pages 91165–91198, 2025.

J. Xiong, Q. Wang, Z. Yang, P. Sun, L. Han, Y. Zheng, H. Fu, T. Zhang, J. Liu, and H. Liu. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv preprint arXiv:1810.06394*, 2018.

J. Xu, V. Makoviychuk, Y. Narang, F. Ramos, W. Matusik, A. Garg, and M. Macklin. Accelerated policy learning with parallel differentiable simulation. *arXiv preprint arXiv:2204.07137*, 2022.

W. Zhang, J. Hu, and A. Abate. On the value functions of the discrete-time switched lqr problem. *IEEE Transactions on Automatic Control*, 54(11):2669–2674, 2009.

W. Zhang, J. Hu, and A. Abate. Infinite-horizon switched lqr problems in discrete time: A suboptimal algorithm with performance analysis. *IEEE Transactions on Automatic Control*, 57(7):1815–1821, 2011.

Organization of the appendix. We divide the appendix into four sections. Appendix A formalizes the problem classes studied in the paper and provides detailed model descriptions. Appendix B contains the theoretical results and their proofs, along with gradient derivations for alternative policy classes. Appendix C describes the policy parameterization and implementation details used in our experiments. Finally, Appendix D provides full experimental specifications, additional results, and details of the gradient quality analysis.

A Problem setup and models

This appendix formalizes the problem classes considered in the paper and provides detailed model descriptions. Appendix A.1 summarizes our notation. Appendix A.2 gives a formal definition of hybrid MDPs. Appendix A.3 defines piecewise-smooth MDPs and shows how they can be represented as hybrid MDPs. Appendix A.4 illustrates this reformulation through an inventory control example with fixed costs and varying lead times. Appendix A.5 presents a complete formulation of the joint replenishment problem, alongside its hybrid MDP representation.

A.1 Notation summary

Table 1 summarizes the main notation used throughout the paper.

A.2 Hybrid discrete-continuous action MDPs

Here, we provide a formal definition of hybrid MDPs. A *hybrid MDP* has state space $\mathcal{S} \subseteq \mathbb{R}^{d_\mathcal{S}}$ for some $d_\mathcal{S} \in \mathbb{N}$, and action space of the form $\mathcal{A} = \bigcup_{j \in \mathcal{X}} \{j\} \times \mathcal{B}^j$, where $\mathcal{X}$ is a finite set of discrete actions, and for each $j \in \mathcal{X}$, $\mathcal{B}^j \subseteq \mathbb{R}^p$ denotes the set of feasible continuous controls. An action at time t is therefore a pair $a_t = (x_t, b_t)$, where $x_t \in \mathcal{X}$ and $b_t \in \mathcal{B}^{x_t}$.

To obtain PW gradients for the continuous action, we impose the following assumption.

Assumption 2 (Smoothness of dynamics and costs). For every discrete action x and exogenous realization ξ, the transition and cost functions $(s, b) \mapsto f(s, (x, b), \xi)$ and $(s, b) \mapsto c(s, (x, b), \xi)$ are continuous and almost everywhere (a.e.) differentiable with respect to (s, b).

Assumption 2 imposes smoothness only with respect to (s, b), but no conditions on the discrete action x. In contrast, applying standard PW methods typically requires problem settings with fully continuous action spaces and global smoothness with respect to all action components.

A.3 Piecewise smooth MDPs

MDPs with a finite set of discontinuities in the action are a key class of problems we wish to address under our framework. We formalize them as Piecewise smooth MDP (PS-MDP) as follows:

Definition 1 (PS-MDP)**.** An MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, c, f, \mathbb{P}_\xi)$ is a piecewise-smooth MDP with J regions if i) the action space admits a partition $\mathcal{A} = \bigcup_{j=1}^{J} \mathcal{B}^j$, where each $\mathcal{B}^j \subseteq \mathbb{R}^p$ is non-empty and connected, and ii) for each $j \in [J]$, the restricted functions

$$c|_{\mathcal{B}^j}(s, a, \xi), \quad f|_{\mathcal{B}^j}(s, a, \xi)$$

are continuous and almost everywhere differentiable with respect to a, for all (s, ξ).

To represent a PS-MDP as a Hybrid MDP and unlock the benefits of MGPO, we exploit the partition of the action space given by Definition 1, augmenting it into a discrete-continuous representation in which the discrete component selects a region and the continuous component specifies a control within that region. Crucially, while the original PS-MDP may be discontinuous in a, the resulting hybrid MDP is smooth in b within each region, satisfying Assumption 2 and thereby enabling PW gradient estimation. We formalize this below, and provide a concrete example in the context of inventory control in Appendix A.4.

Proposition 1. *Every PS-MDP with J regions admits a representation as a hybrid MDP $\tilde{\mathcal{M}}$ satisfying Assumption 2 such that for every policy π in the PS-MDP, there exists a policy π in $\tilde{\mathcal{M}}$ that induces the same sequence of states and costs for every realization ξ, and vice versa.*

Table 1: Summary of notation.

Symbol	Meaning
MDP with exogenous randomness (Section 2.1)	
$s_t \in \mathcal{S}$	State at period t
$a_t \in \mathcal{A}(s_t)$	Action at period t
$\xi = (\xi_0, \dots, \xi_{T-1})$	Sequence of exogenous random terms
$\bar{\xi}$	Scenario (a fixed realization of ξ)
$\mathbb{P}_\xi$	Joint law of ξ
$f(s, a, \xi)$	Transition function, $s_{t+1} = f(s_t, a_t, \xi_t)$
$c(s, a, \xi)$	Cost function
γ, T	Discount factor; horizon
π	Policy
$J(\pi)$	Expected discounted cost under policy π
V_π, Q^π	State-value and action-value functions of policy π
Hybrid MDPs (Section 2.2)	
$a = (x, b)$	Hybrid action: discrete x and continuous b
$\mathcal{X}$	Finite set of discrete actions
p	Continuous action dimension
J	Number of discrete actions
$\mathcal{B}^x \subseteq \mathbb{R}^p$	Feasible continuous actions given discrete action x
Policies and parameters (Section 3)	
$\pi^{\mathcal{X}}$	Discrete policy
$\pi^{\mathcal{B}}$	Continuous policy
ϕ	Parameters of the discrete policy
κ	Parameters of the continuous policy
$\theta = (\phi, \kappa)$	All policy parameters
$\pi^{\mathcal{X}}_\phi$	Parameterized discrete policy
$\pi^{\mathcal{B}}_\kappa$	Parameterized continuous policy
$\pi_\theta = (\pi^{\mathcal{X}}_\phi, \pi^{\mathcal{B}}_\kappa)$	Parameterized hybrid policy
$b_\kappa(s, x) = \pi^{\mathcal{B}}_\kappa(s)[x]$	Continuous action corresponding to discrete action x
$s_t(\kappa)$, $b_t(\kappa)$	State and continuous action along a realized $(\xi, x_{0:\infty})$, as functions of κ
$\mathbb{E}_\theta$	Expectation over $(\xi, x_{0:\infty})$ induced by π_θ
Gradient estimators and cross-term analysis (Sections 3 and 5)	
SF, PW	Score-function and pathwise gradients
Cross term	Term in $\nabla_\kappa J$ capturing how κ shifts discrete-action probabilities
MGPOFull / MGPONoCross	MGPO with / without the cross term
$\pi^{\mathcal{X}}(\kappa)$, π^*_κ	Discrete best response to κ; corresponding hybrid policy
ρ	Initial-state distribution
ϵ-best response	Discrete policy within ϵ of the best response in expected cost

Proof. Define $\tilde{\mathcal{M}}$ with action space

$$\tilde{\mathcal{A}} = \bigcup_{j=1}^{J} \{j\} \times \mathcal{B}^j,$$

transition $f(s, (x, b), \xi) = f^x(s, b, \xi)$ and cost $c(s, (x, b), \xi) = c^x(s, b, \xi)$. Assumption 2 is satisfied by condition ii) of the PS-MDP definition. For any policy π in the PS-MDP, define π by setting $x_t = j$ whenever $\pi_t(s) \in \mathcal{B}^j$, and $b_t = \pi_t(s)$. By construction, $b_t \in \mathcal{B}^{x_t}$ and $f^{x_t}(s, b_t, \xi) = f(s, \pi_t(s), \xi)$, with the same holding for costs, so the induced trajectories are identical. Conversely, for any policy π in $\tilde{\mathcal{M}}$, define π in the PS-MDP by setting $\pi_t(s) = b_t$, where $(x_t, b_t) = \pi_t(s)$. Since $b_t \in \mathcal{B}^{x_t} \subseteq \mathcal{A}$, π is feasible, and the induced trajectories are identical by the same argument. □

A.4 PS-MDP Reformulation: inventory control with fixed costs and varying lead times

We present a short example to show how to represent a PS-MDP with J regions as a hybrid MDP.

Example 3. Consider the setting of a retailer that sells a single product across T periods. The retailer pays a linear underage cost u for unmet demand, and a linear holding cost h for carrying inventory at

the end of each period. The retailer's supplier charges fixed costs depending on the order size, as she needs to consolidate the orders into pallets: 0 for orders between 0 and 5 (as a "promotion", the first pallet is free), 10 for orders between 5 and 10, and 20 for orders between 10 and 20. Procurement costs per unit are negligible. Similarly, the supplier's lead time varies depending on the order size, given that she might need to buy additional input materials: orders between 0 and 7 can be delivered in 2 periods, while orders between 7 and 20 in 3 periods. They agree that all units in an order will be delivered simultaneously.

PS-MDP Representation. We can model this problem as a PS-MDP with $J = 4$ regions as follows. The system state at time t is $s_t = (I_t, Q_t)$, where $I_t \in \mathbb{R}$ is the inventory on-hand and $Q_t \in \mathbb{R}^2_+$ tracks outstanding orders up to one minus the maximum lead time of 3 periods. The action at time t is the order quantity $a_t \in [0, 20]$.

Inventory on hand evolves as

$$I_{t+1} = \max\{I_t - \xi_t, 0\} + Q_t[1],$$

where $Q_t[i]$ is the i-th component of Q_t. The outstanding orders update as

$$Q_{t+1} = \begin{cases} (Q_t[2] + a_t, 0), & \text{if } 0 \le a_t \le 7, \\ (Q_t[2], a_t), & \text{if } 7 < a_t \le 20. \end{cases}$$

Finally, per-period costs are computed as

$$c(s_t, a_t, \xi_t) = u(\xi_t - I_t)^+ + h(I_t - \xi_t)^+ + \text{FixedCost}(a_t),$$

with

$$\text{FixedCost}(a_t) = \begin{cases} 0, & \text{if } 0 \le a_t \le 5, \\ 10, & \text{if } 5 < a_t \le 10, \\ 20, & \text{if } 10 < a_t \le 20. \end{cases}$$

Hybrid MDP Representation. We define the hybrid MDP representation of the MDP described above with $J = 4$ regions. The action space is given by

$$\mathcal{A} = \bigcup_{j=1}^{4} \{j\} \times \mathcal{B}^j,$$

where

$$\mathcal{B}^1 = [0, 5], \quad \mathcal{B}^2 = (5, 7], \quad \mathcal{B}^3 = (7, 10], \quad \mathcal{B}^4 = (10, 20].$$

The transition function is defined as

$$f(s_t, (x_t, b_t), \xi_t) = f^{x_t}(s_t, b_t, \xi_t),$$

where each f^j (which returns a pair (I_{t+1}, Q_{t+1})) is given by

$$f^j(s_t, b_t, \xi_t) = \begin{cases} \left(\max\{I_t - \xi_t, 0\} + Q_t[1],\ (Q_t[2] + b_t, 0)\right), & \text{for } j = 1, 2, \\ \left(\max\{I_t - \xi_t, 0\} + Q_t[1],\ (Q_t[2], b_t)\right), & \text{for } j = 3, 4. \end{cases}$$

Similarly, the cost function is given by

$$c(s_t, (x_t, b_t), \xi_t) = c^{x_t}(s_t, b_t, \xi_t),$$

where

$$c^j(s_t, b_t, \xi_t) = u(\xi_t - I_t)^+ + h(I_t - \xi_t)^+ + K_j,$$

with $K_1 = 0, \quad K_2 = 10, \quad K_3 = 10, \quad K_4 = 20.$

A.5 Detailed Joint Replenishment Problem Formulation

We provide a complete formulation of the joint replenishment problem introduced in Example 1. We consider p products over a finite horizon of T periods. Demand for product k in period t is denoted by ξ_t^k, and we write $\xi_t = (\xi_t^1, \dots, \xi_t^p)$. We assume a lead time $L \geq 2$ to avoid edge cases.

State and action spaces. The system state at time t is

$$s_t = (I_t, Q_t),$$

where $I_t = (I_t^1, \dots, I_t^p) \in \mathbb{R}^p$ denotes inventory levels under backlogged demand, and

$$Q_t = (Q_t^1, \dots, Q_t^p), \qquad Q_t^k \in \mathbb{R}_+^{L-1},$$

tracks outstanding orders for product k that have not yet arrived.

After observing s_t, the retailer chooses an order vector

$$a_t = (a_t^1, \dots, a_t^p) \in \mathbb{R}_+^p,$$

where a_t^k is the order quantity for product k.

Transition dynamics. Inventory evolves according to

$$I_{t+1}^k = I_t^k - \xi_t^k + Q_t^k[1], \qquad k \in [p].$$

Outstanding orders update as

$$Q_{t+1}^k = (Q_t^k[2], \dots, Q_t^k[L-1], a_t^k), \qquad k \in [p].$$

That is, the pipeline is shifted one position forward and the newly placed order is inserted in the last position.

Cost function. The retailer incurs linear underage and holding costs for each product, as well as a shared fixed ordering cost whenever at least one product is ordered. The per-period cost is

$$c(s_t, a_t, \xi_t) = \sum_{k=1}^{p} \left[u_k(\xi_t^k - I_t^k)^+ + h_k(I_t^k - \xi_t^k)^+ \right] + K \cdot \mathbf{1}\left\{ \sum_{k=1}^{p} a_t^k > 0 \right\}.$$

Here, u_k and h_k denote the underage and holding costs for product k, respectively, and K is the shared fixed cost incurred whenever the retailer places a positive order for at least one product.

A.5.1 Hybrid MDP Reformulation for the Joint Replenishment Problem

We show how the joint replenishment problem in Appendix A.5 can be represented as a hybrid MDP. The discontinuity is induced by the fixed cost K, which is incurred whenever at least one product is ordered.

We define a hybrid action space with two discrete modes:

$$\mathcal{X} = \{0, 1\},$$

where $x_t = 0$ denotes "do not order" and $x_t = 1$ denotes "place an order." The corresponding continuous action regions are

$$\mathcal{B}^0 = \{0\}^p, \qquad \mathcal{B}^1 = \mathbb{R}_+^p \setminus \{0\}^p.$$

In practice, we replace $\mathcal{B}^0 = \{0\}^p$ with a small hypercube $\mathcal{B}^0 = [0, \epsilon]^p$ for a small $\epsilon > 0$ to avoid degenerate gradients; this does not affect the conceptual reformulation.

We define

$$f^0(s_t, b_t, \xi_t) = \left(I_t^k - \xi_t^k + Q_t^k[1],\ (Q_t^k[2], \dots, Q_t^k[L-1], 0) \right)_{k \in [p]},$$
$$f^1(s_t, b_t, \xi_t) = \left(I_t^k - \xi_t^k + Q_t^k[1],\ (Q_t^k[2], \dots, Q_t^k[L-1], b_t^k) \right)_{k \in [p]}.$$

The transition is therefore

$$f(s_t, (x_t, b_t), \xi_t) = f^{x_t}(s_t, b_t, \xi_t),$$

with $b_t \in \mathcal{B}^{x_t}$.

Similarly, cost branches are given by

$$c^0(s_t, b_t, \xi_t) = \sum_{k=1}^{p} \left[u_k(\xi_t^k - I_t^k)^+ + h_k(I_t^k - \xi_t^k)^+ \right],$$

and

$$c^1(s_t, b_t, \xi_t) = \sum_{k=1}^{p} \left[u_k(\xi_t^k - I_t^k)^+ + h_k(I_t^k - \xi_t^k)^+ \right] + K.$$

The cost is

$$c(s_t, (x_t, b_t), \xi_t) = c^{x_t}(s_t, b_t, \xi_t).$$

B Theoretical underpinnings

This appendix provides the theoretical results underlying the proposed framework. First, Appendix B.1 defines notation and assumptions used throughout this appendix. Appendix B.2 derives the mixed gradient estimator for hybrid policies. Appendix B.3 proves that the cross term vanishes near a discrete best response. Finally, Appendix B.4 presents the SF estimator for fully stochastic policies.

B.1 Additional notation and assumptions

Throughout this section, we consider a Hybrid MDP as defined in Appendix A.2 satisfying Assumption 2.

Notation. We introduce notation to make explicit the dependence of trajectories on the policy parameters, which will be key for gradient computation. For $t_1 \leq t_2$, we write $x_{t_1:t_2} = (x_{t_1}, \ldots, x_{t_2})$ (and similarly for $\xi_{t_1:t_2}$) to denote a subsequence. We consider a stochastic process in which $\theta = (\phi, \kappa)$ induces a joint distribution $\mathcal{D}_\theta$ over $(\xi_{0:\infty}, x_{0:\infty})$, and write $\mathbb{E}_\theta[\cdot] = \mathbb{E}_{(\xi_{0:\infty}, x_{0:\infty}) \sim \mathcal{D}_\theta}[\cdot]$.

We use parentheses to indicate parameter dependence only when that dependence remains *after realization of the process* $(\xi_{0:\infty}, x_{0:\infty})$. In particular, κ influences the distribution $\mathcal{D}_\theta$, while, for a fixed realization $(\xi_{0:\infty}, x_{0:\infty}) \sim \mathcal{D}_\theta$, the state trajectory is a deterministic function of the same κ (now viewed as a variable argument), which we denote by $s_t(\kappa)$. Conditional on this realization, $s_t(\kappa)$ does not depend on ϕ, since the discrete actions $x_{0:\infty}$ are fixed. This conditioning highlights the deterministic dependence on κ along a realized trajectory, while the distribution of $(\xi_{0:\infty}, x_{0:\infty})$ remains parameter-dependent. We write $b_t(\kappa)$ for the continuous action executed at time t, defined as $b_t(\kappa) = b_\kappa(s_t(\kappa), x_t)$, with dependence on x_t implicit. Accordingly, we write costs as $c_t(\kappa) = c\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big)$ and transitions as $s_{t+1}(\kappa) = f\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big)$.

For a fixed initial state s_0, the expected cost is

$$J(\theta) = \mathbb{E}_\theta \left[\sum_{t=0}^{\infty} \gamma^t c_t(\kappa) \,\middle|\, s_0 \right], \tag{3}$$

and we denote by $Q_{\pi_\theta}(s, a)$ the corresponding action-value function.

Regularity conditions. The proof of Theorem 1 requires the following conditions on the policy class, which ensure the gradient estimator is well-defined and allow interchange of differentiation and expectation.

Assumption 3 (Policy regularity). For each $\theta = (\phi, \kappa)$:

1. The discrete policy $s \mapsto \pi_\phi^{\mathcal{X}}(x \mid s)$ is continuous and a.e. differentiable for every $x \in \mathcal{X}$.
2. The continuous policy mapping $\kappa \mapsto b_\kappa(s, x)$ is continuous and a.e. differentiable for every $s \in \mathcal{S}, x \in \mathcal{X}$.
3. The following gradients exist a.e. and are uniformly bounded over t, $\xi_{0:t}$ and $x_{0:t} \in \mathcal{X}^{t+1}$: $\nabla_\phi \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))$, $\nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))$, and $\nabla_\kappa c(s_t(\kappa), (x_t, b_\kappa(s_t(\kappa), x_t)), \xi_t)$.
4. The cost function $c(s, a, \xi)$ is uniformly bounded.

These conditions ensure the gradient estimator is well-defined and allow interchange of differentiation and expectation.

B.2 Proof of Theorem 1

Proof. Under Assumption 3, the mapping $s \mapsto \pi_\phi^{\mathcal{X}}(x \mid s)$ is a.e. differentiable with bounded log-gradients, which ensures that the SF estimator is well-defined. Therefore, by the standard policy gradient theorem [Sutton et al., 1999], we have

$$\nabla_\phi J(\theta) = \mathbb{E}_\theta\left[\sum_{t=0}^{\infty} \gamma^t \, \nabla_\phi \log \pi_\phi^{\mathcal{X}}\left(x_t \mid s_t(\kappa)\right) \; Q_{\pi_\theta}\big(s_t(\kappa), (x_t, b_t(\kappa))\big)\right]. \tag{4}$$

We now turn to deriving the gradient with respect to the continuous parameters κ. The derivation follows the standard approach of differentiating expectations over trajectories, and is closely related to the gradient estimators derived in the stochastic computation graph framework of Schulman et al. [2015a].

For a fixed initial state s_0, we first derive the gradient of the undiscounted cost at time t with respect to κ:

$$(1) \;=\; \nabla_\kappa \, \mathbb{E}_\theta\big[c(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t) \mid s_0\big]. \tag{5}$$

Expanding the nested expectations and interchanging differentiation and integration (which is justified by Assumptions 2 and 3, which ensure that the relevant derivatives exist almost everywhere, the gradients and costs are uniformly bounded, and hence dominated convergence applies), we obtain

$$\begin{aligned}(1) &= \nabla_\kappa \sum_{x_{0:t} \in \mathcal{X}^{t+1}} \int p(\xi_{0:t}) \left(\prod_{u=0}^{t} \pi_\phi^{\mathcal{X}}\big(x_u \mid s_u(\kappa)\big)\right) c\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big) \, d\xi_{0:t} \\ &= \sum_{x_{0:t} \in \mathcal{X}^{t+1}} \int p(\xi_{0:t}) \nabla_\kappa \left[\left(\prod_{u=0}^{t} \pi_\phi^{\mathcal{X}}\big(x_u \mid s_u(\kappa)\big)\right) c\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big)\right] \, d\xi_{0:t},\end{aligned}$$

where $p(\xi_{0:t})$ does not depend on (ϕ, κ).

Applying the product rule, the integrand decomposes into two terms.

For the first term,

$$\begin{aligned}(2) &= \left[\nabla_\kappa \left(\prod_{u=0}^{t} \pi_\phi^{\mathcal{X}}\big(x_u \mid s_u(\kappa)\big)\right)\right] \cdot c\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big) \\ &= \left[\sum_{v=0}^{t} \left(\prod_{u=0}^{t} \pi_\phi^{\mathcal{X}}\big(x_u \mid s_u(\kappa)\big) \frac{\nabla_\kappa \pi_\phi^{\mathcal{X}}\big(x_v \mid s_v(\kappa)\big)}{\pi_\phi^{\mathcal{X}}\big(x_v \mid s_v(\kappa)\big)}\right)\right] \cdot c\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big) \\ &= \left(\prod_{u=0}^{t} \pi_\phi^{\mathcal{X}}\big(x_u \mid s_u(\kappa)\big)\right) \left[\sum_{v=0}^{t} \nabla_\kappa \log \pi_\phi^{\mathcal{X}}\big(x_v \mid s_v(\kappa)\big)\right] \cdot c\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big).\end{aligned} \tag{6}$$

For the second term,

$$(3) = \left(\prod_{u=0}^{t} \pi_\phi^{\mathcal{X}}\big(x_u \mid s_u(\kappa)\big)\right) \cdot \nabla_\kappa \, c\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big). \tag{7}$$

Combining (6) and (7), we obtain

$$(1) = \mathbb{E}_\theta\left[\sum_{v=0}^{t} \nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_v \mid s_v(\kappa)) \; c\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big) \; + \; \nabla_\kappa c\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big)\right].$$

Summing over t incorporating the discount factor γ^t and using linearity of expectation,

$$\begin{aligned}\nabla_\kappa J(\phi, \kappa) = \mathbb{E}_\theta\Bigg[&\sum_{t=0}^{\infty} \sum_{v=0}^{t} \gamma^t \, \nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_v \mid s_v(\kappa)) \; c\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big) \\ &+ \sum_{t=0}^{\infty} \gamma^t \, \nabla_\kappa c\big(s_t(\kappa), (x_t, b_t(\kappa)), \xi_t\big)\Bigg].\end{aligned}$$

Swapping the order of summation,

$$\nabla_\kappa J(\theta) = \mathbb{E}_\theta\Bigg[\sum_{v=0}^{\infty}\gamma^{\,v}\,\nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_v \mid s_v(\kappa))\sum_{t=v}^{\infty}\gamma^{t-v}c\big(s_t(\kappa),(x_t,b_t(\kappa)),\xi_t\big) + \sum_{t=0}^{\infty}\gamma^{\,t}\,\nabla_\kappa c\big(s_t(\kappa),(x_t,b_t(\kappa)),\xi_t\big)\Bigg]$$

$$= \mathbb{E}_\theta\left[\sum_{t=0}^{\infty}\gamma^{\,t}\,\nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))\; Q_{\pi_\theta}(s_t(\kappa),a_t) + \sum_{t=0}^{\infty}\gamma^{\,t}\,\nabla_\kappa c\big(s_t(\kappa),(x_t,b_t(\kappa)),\xi_t\big)\right],$$

where the final equality follows from the tower property, since the conditional expectation of $\sum_{t=v}^{\infty}\gamma^{t-v}c(s_t(\kappa),(x_t,b_t(\kappa)),\xi_t)$ given $(s_v(\kappa),a_v)$ is $Q_{\pi_\theta}(s_v(\kappa),a_v)$.

□

B.3 Proof of Theorem 2

Proof of Theorem 2. For a fixed continuous policy parameter κ, define the post-discrete action-value function

$$\tilde{Q}_\pi(s,x) := Q_\pi\big(s,(x,\pi_\kappa^{\mathcal{B}}(s,x))\big),$$

that is, the value of selecting discrete action x and then applying the continuous action prescribed by $\pi_\kappa^{\mathcal{B}}$ conditional on x. Similarly, define the post-discrete advantage of the best-response policy π_κ^* as

$$\tilde{A}_{\pi_\kappa^*}(s,x) := \tilde{Q}_{\pi_\kappa^*}(s,x) - V_{\pi_\kappa^*}(s).$$

Since π_κ^* is a statewise best response by Assumption 1(1), we have

$$\tilde{A}_{\pi_\kappa^*}(s,x) \geq 0 \qquad \text{for all } (s,x).$$

By the performance difference lemma for costs,

$$\mathbb{E}_{s_0\sim\rho}\big[V_{\pi_\theta}(s_0) - V_{\pi_\kappa^*}(s_0)\big] = \mathbb{E}_\theta\left[\sum_{t\geq 0}\gamma^t \tilde{A}_{\pi_\kappa^*}(s_t(\kappa),x_t)\right].$$

Since $\pi_\phi^{\mathcal{X}}$ is an ϵ-best response to κ, the left-hand side is at most ϵ, and therefore

$$\mathbb{E}_\theta\left[\sum_{t\geq 0}\gamma^t \tilde{A}_{\pi_\kappa^*}(s_t(\kappa),x_t)\right] \leq \epsilon. \tag{8}$$

Let

$$\mathcal{C}_{\text{cross}} := \mathbb{E}_\theta\left[\sum_{t\geq 0}\gamma^t \nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))\, Q_{\pi_\theta}(s_t(\kappa),a_t)\right]$$

denote the cross term. Since

$$Q_{\pi_\theta}(s_t(\kappa),a_t) = \tilde{Q}_{\pi_\theta}(s_t(\kappa),x_t),$$

we can write

$$\begin{aligned}\mathcal{C}_{\text{cross}} &= \mathbb{E}_\theta\left[\sum_{t\geq 0}\gamma^t \nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))\, \tilde{Q}_{\pi_\theta}(s_t(\kappa),x_t)\right] \\ &= \mathbb{E}_\theta\left[\sum_{t\geq 0}\gamma^t \nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))\, \tilde{Q}_{\pi_\kappa^*}(s_t(\kappa),x_t)\right] \\ &\quad + \mathbb{E}_\theta\left[\sum_{t\geq 0}\gamma^t \nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))\, \Big(\tilde{Q}_{\pi_\theta}(s_t(\kappa),x_t) - \tilde{Q}_{\pi_\kappa^*}(s_t(\kappa),x_t)\Big)\right].\end{aligned}$$

For fixed realized history up to time t excluding x_t, the state $s_t(\kappa)$ is fixed as a differentiable function of κ. Hence

$$\sum_{x\in\mathcal{X}} \pi_\phi^{\mathcal{X}}(x \mid s_t(\kappa))\nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x \mid s_t(\kappa)) = \nabla_\kappa \sum_{x\in\mathcal{X}} \pi_\phi^{\mathcal{X}}(x \mid s_t(\kappa)) = 0.$$

Thus, any term depending only on $s_t(\kappa)$ may be subtracted from $\tilde{Q}_{\pi_\kappa^*}(s_t(\kappa), x_t)$ inside the conditional expectation over x_t. Therefore,

$$\mathcal{C}_{\text{cross}} = T_1 + T_2,$$

where

$$T_1 := \mathbb{E}_\theta\left[\sum_{t\geq 0} \gamma^t \nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))\, \tilde{A}_{\pi_\kappa^*}(s_t(\kappa), x_t)\right],$$

and

$$T_2 := \mathbb{E}_\theta\left[\sum_{t\geq 0} \gamma^t \nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa)) \left(\tilde{Q}_{\pi_\theta}(s_t(\kappa), x_t) - \tilde{Q}_{\pi_\kappa^*}(s_t(\kappa), x_t)\right)\right].$$

By Assumption 1(3),

$$\left\|\nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))\right\| \leq \left\|\nabla_s \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))\right\| \left\|\nabla_\kappa s_t(\kappa)\right\| \leq L_\pi L_s.$$

Using this bound and (8),

$$\begin{aligned}
\|T_1\| &\leq \mathbb{E}_\theta\left[\sum_{t\geq 0} \gamma^t \left\|\nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))\right\| \tilde{A}_{\pi_\kappa^*}(s_t(\kappa), x_t)\right] \\
&\leq L_\pi L_s\, \mathbb{E}_\theta\left[\sum_{t\geq 0} \gamma^t \tilde{A}_{\pi_\kappa^*}(s_t(\kappa), x_t)\right] \\
&\leq L_\pi L_s\, \epsilon.
\end{aligned}$$

It remains to bound T_2. For any (s_t, x_t), both $\tilde{Q}_{\pi_\theta}(s_t(\kappa), x_t)$ and $\tilde{Q}_{\pi_\kappa^*}(s_t(\kappa), x_t)$ use the same immediate cost and the same continuous action $\pi_\kappa^{\mathcal{B}}(s_t(\kappa), x_t)$ in the first period. They differ only in the continuation policy. Hence,

$$\begin{aligned}
\tilde{Q}_{\pi_\theta}(s_t(\kappa), x_t) - \tilde{Q}_{\pi_\kappa^*}(s_t(\kappa), x_t) &= \gamma\, \mathbb{E}\big[V_{\pi_\theta}(s_{t+1}) - V_{\pi_\kappa^*}(s_{t+1}) \mid s_t(\kappa), x_t\big] \\
&\leq \gamma C\epsilon,
\end{aligned} \tag{9}$$

where the inequality follows from Assumption 1(2) applied to the conditional distribution of s_{t+1} given $(s_t(\kappa), x_t)$. Moreover, the left-hand side is nonnegative by the statewise optimality of π_κ^*.

Using (9) and Assumption 1(3),

$$\begin{aligned}
\|T_2\| &\leq \mathbb{E}_\theta\left[\sum_{t\geq 0} \gamma^t \left\|\nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))\right\| \left|\tilde{Q}_{\pi_\theta}(s_t(\kappa), x_t) - \tilde{Q}_{\pi_\kappa^*}(s_t(\kappa), x_t)\right|\right] \\
&\leq L_\pi L_s \mathbb{E}_\theta\left[\sum_{t\geq 0} \gamma^t \gamma C\epsilon\right] \\
&= L_\pi L_s\, \frac{C\gamma}{1-\gamma}\, \epsilon.
\end{aligned}$$

Combining the bounds for T_1 and T_2 gives

$$\|\mathcal{C}_{\text{cross}}\| \leq L_\pi L_s\, \epsilon \left(1 + \frac{C\gamma}{1-\gamma}\right),$$

which proves the result. □

B.4 Score-function estimator for fully stochastic policies

In this section, we consider a fully stochastic version of the policy, in which the continuous component is also stochastic. We represent the policy as first sampling a discrete action $x_t \sim \pi_\phi^{\mathcal{X}}(\cdot \mid s_t)$, and then sampling the corresponding continuous action as $b_t \sim \pi_\kappa^{\mathcal{B}}(\cdot \mid s_t, x_t)$. Therefore,

$$\pi_\theta(a_t \mid s_t) = \pi_\phi^{\mathcal{X}}(x_t \mid s_t)\, \pi_\kappa^{\mathcal{B}}(b_t \mid s_t, x_t).$$

Under Assumption 3, and assuming that the continuous policy admits a density with respect to Lebesgue measure, whose log-density is almost everywhere differentiable with bounded gradients, the standard SF estimator applies [Sutton et al., 1999]. In particular,

$$\nabla_\theta J(\theta) = \mathbb{E}_\theta \left[\sum_{t=0}^{\infty} \gamma^t \, \nabla_\theta \log \pi_\theta(a_t \mid s_t)\; Q_{\pi_\theta}(s_t, a_t) \right]. \tag{10}$$

C Policy class and implementation

This appendix provides additional details on our policy class and algorithm implementation. Appendix C.1 details the architecture of the hybrid policy, including its discrete and continuous components and feasibility enforcement mechanisms. Appendix C.2 presents the full MGPO procedure used in our experiments (Algorithm 1), along with implementation details and hyperparameters used across experimental settings.

C.1 Policy Parameterization

For all our experiments, we consider the policy architecture illustrated in Figure 4. The policy is composed of a discrete component and a continuous component, each represented by a neural network.

The discrete policy $\pi_\phi^{\mathcal{X}}$ maps the state s_t to a vector of logits over discrete actions. A softmax transformation is applied to obtain a probability distribution, from which a discrete action x_t is sampled.

The continuous policy $\pi_\kappa^{\mathcal{B}}$ maps the state s_t to a vector of candidate continuous actions $(b_t^1, \ldots, b_t^J)$, one for each discrete action. Given the sampled discrete action x_t, the executed continuous action is selected as $b_t = b_t^{x_t}$. This parameterization is equivalent in expressive power to defining a conditional continuous policy that takes (s_t, x_t) as input and outputs a single continuous action. We experimented with such a conditional design and observed similar empirical performance.

Differentiable feasibility enforcement. We enforce feasibility of the continuous actions through simple output transformations. For the JRP setting (see Example 1), actions are required to be non-negative, and we therefore apply a softplus transformation to the network outputs. More complex constraints can be enforced as well—for instance, simplex or budget constraints via softmax-type normalizations, or interval constraints via sigmoid-based transformations—but are beyond the scope of this work.

C.2 Implementation Details

We provide additional details on the implementation of MGPO. Algorithm 1 presents the full procedure used in our experiments. The remainder of this section summarizes the implementation choices and hyperparameters used across all experiments, unless noted otherwise.

Hybrid optimization scheme. We optimize the discrete and continuous components of the policy using different gradient estimators. The discrete policy parameters are updated using the gradient in (1), implemented with PPO-style clipped objectives as described below.

For the continuous parameters, **MGPOFull** uses the full mixed gradient estimator in (2), while **MGPONoCross** uses only the PW component of this estimator. For all SF terms (including the cross term), we apply standard PPO modifications, replacing Q-values with advantage estimates derived from a learned value function.

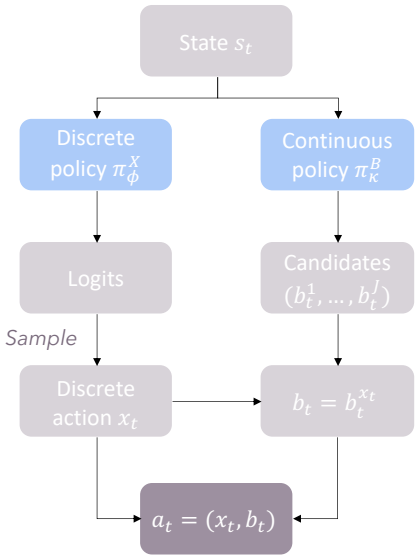


Figure 4: Policy architecture for hybrid MDPs.

Algorithm 1 Mixed Gradient Policy Optimization with PPO-style Updates

Initialize policy parameters $\theta = (\phi, \kappa)$ and value parameters ψ
Let $\mathcal{D} = \{\xi_{0:T-1}^h\}_{h=1}^H$ denote the fixed training dataset of exogenous scenarios
for iteration $= 1, 2, \ldots,$ `max_iterations` **do**
    Partition $\mathcal{D}$ into training batches
    for each training batch $\mathcal{H} \subset \mathcal{D}$ **do**
        Roll out the policy $\pi_\theta = (\pi_\phi^{\mathcal{X}}, \pi_\kappa^{\mathcal{B}})$ on $\mathcal{H}$
        Compute costs, value estimates V_ψ, and advantages $\widehat{A}_t$ using GAE

        Continuous update
        Take one step on κ using all trajectories in $\mathcal{H}$:
    – **MGPOFull**: the full mixed estimator in (2), with SF terms weighted by $\widehat{A}_t$;
    – **MGPONoCross**: only the PW component of (2).

        Discrete PPO updates
        for PPO epoch $= 1, \ldots,$ `max_ppo_epochs` **do**
            Partition $\mathcal{H}$ into `num_minibatches` minibatches
            for each minibatch **do**
                Update ϕ using the PPO clipped objective based on (1)
                Update value parameters ψ
            end for
            Stop PPO updates early if the approximate KL divergence exceeds the target threshold
        end for
    end for
end for

For the H-PPO baseline, we use the SF gradient in (10) with the same PPO-style modifications.

Computation of gradient terms. We implement a differentiable hybrid simulator in PyTorch, enabling PW gradients through the continuous components via automatic differentiation. For the PW component, we differentiate the realized trajectory cost with respect to κ, treating the sampled discrete actions as fixed (i.e., no differentiation through the sampling operation). The discrete gradient is computed using SF terms, by differentiating $\log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))$ and weighting by advantage estimates. For the cross term in (2), we compute $\nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))$ via automatic differentiation through the state trajectory, and multiply it by the corresponding advantage estimate.

Optimization. We use Adam with learning rate 10^{-3} and $\epsilon = 10^{-5}$. Gradients are clipped to have maximum norm 5, applied independently to each network. In practice, for computational efficiency, each batch first receives a full-batch policy update to all parameters, even though Algorithm 1 presents the discrete and continuous updates separately. We then perform additional PPO epochs in which the data is split into 4 minibatches and only the discrete policy and value function are updated. For H-PPO [Fan et al., 2019], we follow the CleanRL implementation [Huang et al., 2022b], and thus perform the same number of gradient steps for the continuous and discrete parameters. In Appendix D.4 we consider variants of this scheme to more closely mirror MGPO's update structure, but find that they degrade performance. Additionally, for H-PPO we consider a smaller learning rate of 10^{-4} since larger learning rates lead to unstable performance.

Advantage estimation. Advantages are computed using generalized advantage estimation (GAE) [Schulman et al., 2015b] with $\gamma = 0.99$ and $\lambda = 0.96$. Advantages are normalized to zero mean and unit variance within each minibatch.

Policy and value losses. We consider a clipping parameter of 0.15 for the PPO-style clipped objectives. The value function is trained using a squared-error loss, and weighted by a coefficient of 0.15. An entropy bonus with initial coefficient 0.5 is applied to the discrete policy, which is decayed linearly through training.

Network architecture and initialization. All networks use two hidden layers of size 512 with $\tanh$ activations. Weights are initialized orthogonally, with gain $\sqrt{2}$ for hidden layers, smaller gains for policy outputs, and gain 1.0 for the value head.

Normalization. Costs are standardized by dividing by the batch standard deviation, without centering. For the JRP, state inputs are normalized using a single exponentially weighted estimate of mean demand, computed by averaging demand observations across products and trajectories, with smoothing parameter $\beta = 0.99$; continuous-action outputs are rescaled accordingly.

Early stopping. Training on a batch is terminated early if the approximate KL divergence exceeds a predefined threshold of 0.015.

Compute resources. All experiments were run on NVIDIA A40 GPUs (46 GB memory each), using CUDA 12.2. Each independent trial was run on a single GPU. Most runs completed in under 60 minutes, with all runs completing in under 90 minutes. These figures are approximate, as many experiments were run concurrently on a shared cluster without optimizing for wall-clock time.

D Specifications and results for numerical experiments

This appendix provides full specifications for the numerical experiments in Section 4, along with additional results and hyperparameter details. Appendix D.1 covers the JRP setting, while Appendix D.2 covers the S-LQR setting, including a validation against Riccati-equation baselines. Appendix D.3 provides the full specification of the gradient-quality experiments of Section 4.2, including policy construction, gradient computation, metric definitions, aggregation procedures, and additional results. Finally, Appendix D.4 reports a hyperparameter robustness study for H-PPO, showing that MGPO's advantage persists across a range of configurations.

D.1 JRP setting (Example 1)

Setting specifications. We consider Example 1 with heterogeneous products. Each product's underage cost, holding cost, and mean demand are sampled independently and uniformly from $[6.3, 11.7]$, $[0.7, 1.3]$, and $[6.0, 14.0]$, respectively, and are fixed across trajectories. Lead times are set to $L = 2$ periods for all products. Demand follows a Poisson distribution with mean given by the sampled value. We set the fixed cost to $K = 64p$ to keep the per-product fixed cost constant, and vary the number of products $p \in \{1, 3, 5, 10, 20, 30, 40, 50, 60\}$. We consider trajectories of 100 periods and report per-product per-period costs, omitting the first and last 20 periods to approximate the long-run average cost in a stationary system. The initial inventory state is set to zero. We consider batch sizes of $B \in \{16, 64\}$ trajectories and fix the total number of policy updates to 1600 by scaling

the number of full-dataset training iterations as $1600 \cdot B/1024$. We run 10 independent trials per configuration.

Additional specifications. The policy input is the product inventory vector, normalized by a scalar mean demand estimate computed via an exponentially weighted moving average with parameter 0.99, shared across all products and trajectories. The mean demand estimate and the fixed cost normalized by it are also provided as inputs; while not strictly necessary under i.i.d. demand with a fixed cost, we do so for generality. Network outputs are rescaled by the mean demand estimate before being applied to the environment. Cost parameters (underage and holding costs) are not included as inputs.

Results. Tables 2 and 3 report performance metrics for **MGPOFull**, **MGPONoCross**, and H-PPO across product counts, for batch sizes of 16 and 64 trajectories, respectively. All metrics consider performance on the test set. All three algorithms perform comparably for $p \leq 20$. Beyond this, the effect of batch size becomes pronounced. For $B = 16$, H-PPO degrades sharply from $p = 30$ onward, with mean losses roughly doubling relative to MGPO by $p = 50$ and standard deviations reaching up to over 5% of the mean, reflecting increasing instability. For $B = 64$, H-PPO remains competitive through $p = 40$ but deteriorates at $p = 50$. For $p = 60$, H-PPO's mean loss is roughly 45% higher than that of **MGPOFull**. In contrast, both MGPO variants maintain stable performance across all product counts and both batch sizes.

Comparing **MGPOFull** and **MGPONoCross**, differences are small and inconsistent across most settings, with neither variant consistently dominating the other. The most notable discrepancy appears at $B = 64$, $p = 60$, where **MGPONoCross**'s mean exceeds that of **MGPOFull** by roughly 6%, though both remain far below H-PPO. Overall, the two variants perform similarly, suggesting that the cross term typically provides limited additional signal in this setting.

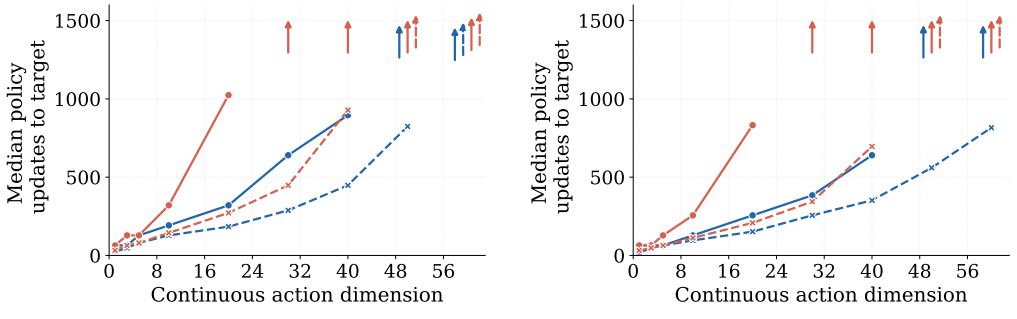
Table 2: JRP test loss across product counts for $B = 16$ trajectories per batch. Mean $\pm$ std and median over 10 independent trials.

	MGPOFull		**MGPONoCross**		**H-PPO**	
p	**Mean $\pm$ Std**	**Median**	**Mean $\pm$ Std**	**Median**	**Mean $\pm$ Std**	**Median**
1	34.96 ± 0.04	34.97	34.97 ± 0.05	34.95	34.94 ± 0.03	34.93
3	37.24 ± 0.08	37.21	37.27 ± 0.07	37.29	37.18 ± 0.06	37.20
5	38.82 ± 0.12	38.80	38.77 ± 0.16	38.76	38.56 ± 0.05	38.56
10	38.79 ± 0.10	38.83	38.85 ± 0.16	38.86	38.68 ± 0.07	38.67
20	39.08 ± 0.12	39.09	39.26 ± 0.16	39.24	39.23 ± 0.08	39.22
30	39.43 ± 0.38	39.33	39.54 ± 0.27	39.53	51.67 ± 2.87	51.85
40	40.40 ± 0.72	40.28	40.82 ± 0.49	40.98	70.37 ± 1.92	69.73
50	43.47 ± 2.19	42.77	42.95 ± 1.43	42.44	81.36 ± 3.27	80.18
60	50.28 ± 3.12	49.71	49.92 ± 1.75	50.39	89.29 ± 2.76	88.59

Table 3: JRP test loss across product counts for $B = 64$ trajectories per batch. Mean $\pm$ std and median over 10 independent trials.

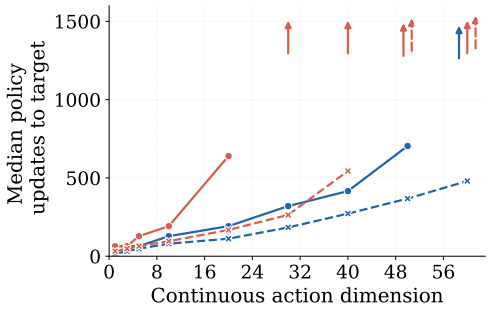
	MGPOFull		**MGPONoCross**		**H-PPO**	
p	**Mean $\pm$ Std**	**Median**	**Mean $\pm$ Std**	**Median**	**Mean $\pm$ Std**	**Median**
1	34.94 ± 0.02	34.93	34.95 ± 0.03	34.96	34.93 ± 0.02	34.94
3	37.09 ± 0.04	37.10	37.10 ± 0.04	37.09	37.09 ± 0.03	37.09
5	38.56 ± 0.08	38.58	38.54 ± 0.06	38.52	38.45 ± 0.02	38.45
10	38.60 ± 0.04	38.61	38.70 ± 0.13	38.67	38.52 ± 0.02	38.52
20	38.83 ± 0.04	38.82	38.97 ± 0.31	38.86	38.85 ± 0.01	38.86
30	38.46 ± 0.06	38.44	38.82 ± 0.32	38.74	38.51 ± 0.03	38.50
40	38.87 ± 0.29	38.76	39.65 ± 0.52	39.48	38.80 ± 0.02	38.80
50	39.66 ± 0.71	39.97	40.06 ± 0.49	39.94	48.75 ± 2.20	48.89
60	40.70 ± 0.98	40.34	43.14 ± 3.07	41.90	59.18 ± 1.62	59.09

Then, Figure 5 reports the median number of policy updates required to reach a target validation performance gap, for batch sizes $B = 16$ and $B = 64$ and target gaps ranging from 5% to 30%. For a fixed batch size, MGPO consistently reaches the target in fewer updates than H-PPO, with the gap widening as the continuous action dimension p grows. When comparing across batch sizes,

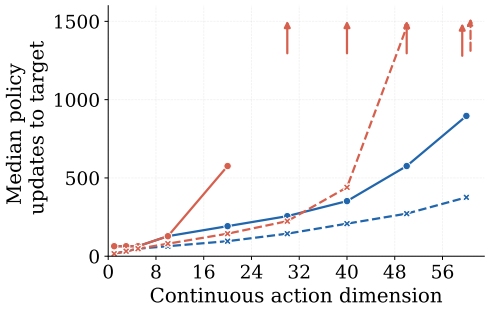


(a) Target gap: 5%.

(b) Target gap: 10%.



(c) Target gap: 20%.



(d) Target gap: 30%.

Figure 5: Median number of policy updates required to reach a target validation-performance gap in the joint replenishment problem, as the continuous action dimension p varies. Each panel uses a different target gap. Blue curves correspond to MGPO and red curves to H-PPO. Solid lines use batch size $B = 16$ trajectories, while dashed lines use batch size $B = 64$ trajectories. Arrows indicate that the median run did not converge for the given configuration.

H-PPO with $B = 64$ is faster than MGPO with $B = 16$ at small p, as the larger batch compensates for H-PPO's weaker gradient estimates. However, H-PPO's convergence speed degrades more steeply with p, and for $p \geq 40$ MGPO with $B = 16$ converges faster than H-PPO with $B = 64$. For the stricter targets (5% and 10%), H-PPO frequently fails to reach the target within the training budget at larger p, as indicated by the truncated curves, while MGPO converges more reliably, especially for $B = 64$.

D.2 S-LQR setting (Example 2)

Setting specifications. We consider Example 2 with matrices constructed so that the system is unstable in the absence of a control. The state dimension and continuous action dimension are set to a common value $p \in \{1, 3, 5, 7, 8, 10, 13, 16, 20, 24, 32, 40, 48\}$, and the number of modes to $J \in \{2, 4, 6, 8\}$. We consider trajectories of 20 periods, as the control typically drives the state to a negligible-cost regime well within this horizon, and report total cost over the 20 periods. We consider 500 training iterations with a batch size of 128 trajectories, corresponding to $500 \cdot 1024/128 = 4000$ policy updates. We run 10 independent trials per configuration.

For each configuration, we generate a shared dynamics matrix $A \in \mathbb{R}^{p \times p}$ and mode-dependent control matrices $\{B_j\}_{j=1}^{J}$ as follows. The dynamics matrix is defined as

$$A = 1.10\, I + 0.05\, N,$$

where $I \in \mathbb{R}^{p \times p}$ is the identity matrix and $N \in \mathbb{R}^{p \times p}$ is a symmetric random matrix constructed by sampling entries i.i.d. from $\mathcal{N}(0, 1)$, symmetrizing via $N \leftarrow (N + N^\top)/2$, and normalizing so that $\max_{i,j} |N_{ij}| = 1$. The scalar $1.10 > 1$ ensures instability in the absence of a control, while the perturbation $0.05\, N$ introduces coupling across state dimensions without significantly altering the overall dynamics. The state coordinates are partitioned into J disjoint groups $\{G_j\}_{j=1}^{J}$ of

approximately equal size. For each mode j, the control matrix $B_j \in \mathbb{R}^{p\times p}$ is diagonal with entries

$$(B_j)_{kk} = \begin{cases} 1.0, & \text{if } k \in G_j, \\ 0.15, & \text{otherwise,} \end{cases}$$

ensuring that each mode provides strong control over a distinct subset of state dimensions and weaker control elsewhere, inducing state-dependent optimal mode selection while permitting controls that can stabilize the system. The cost matrices are fixed across modes and given by $Q_j = I$ and $R_j = 0.1\, I$ for all j, corresponding to a quadratic penalty on state deviation and a relatively low penalty on control effort. This encourages aggressive stabilization and ensures that problem difficulty scales primarily with p. Each state coordinate is initialized independently and uniformly in $[-10/\sqrt{p},\, 10/\sqrt{p}]$ and reset independently across trajectories, keeping the expected squared distance from the origin constant across dimensions.

Additional specifications. The raw state vector is passed directly as input to all networks without normalization, since the initial state scaling already keeps input magnitudes comparable across dimensions.

Results. Table 4 reports performance metrics for **MGPOFull**, **MGPONoCross**, and H-PPO across continuous action dimensions p and mode counts J. Both MGPO variants consistently outperform H-PPO across all configurations, with the gap generally growing as p increases. This pattern holds across all values of J, confirming that the continuous action dimension is the key driver of performance differences. For instance, at $p = 48$, $J = 4$, H-PPO achieves a mean loss of 119.01 against **MGPOFull**'s 76.15, a gap of roughly 56%, and for $J \in \{6, 8\}$ H-PPO diverges, with mean losses above 9000 against below 95 for **MGPOFull**. H-PPO's standard deviations are low at small p but grow substantially where it gets stuck or diverges, indicating unstable convergence. In contrast, both MGPO variants maintain low variance and lower losses across all p and J, highlighting the practical benefits of the mixed gradient estimator in high-dimensional continuous action spaces.

Comparing **MGPOFull** and **MGPONoCross**, dropping the cross term provides a consistent but modest benefit that grows with p. At small dimensions ($p \leq 5$), the two variants perform nearly identically across all J values, with relative differences below 0.5%. **MGPONoCross** achieves a lower mean loss in every configuration, and the gap typically widens for larger p and, to a lesser extent, with J: at $p = 48$, **MGPONoCross** outperforms **MGPOFull** by roughly 0.6–1.9 units across all J, representing a relative improvement of approximately 1.2%–2.1%. While these differences are small in absolute terms, their consistency across all J values and their systematic growth with p suggest that the cross term becomes increasingly detrimental as the continuous action space grows, which we conjecture is due to increased variance.

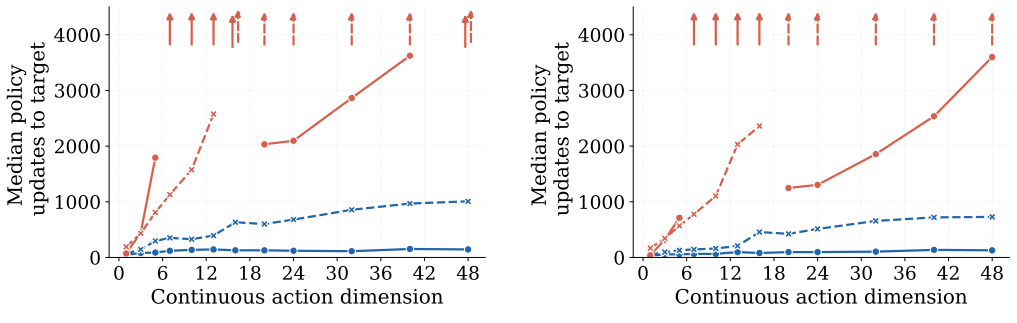
Then, Figures 6 and 7 report the median number of policy updates to reach target gaps ranging from 5% to 30%, for $J \in \{2, 8\}$ and $J \in \{4, 6\}$, respectively. The performance gap between MGPO and H-PPO is drastic and consistent across all values of J: H-PPO requires substantially more updates than MGPO at moderate p, and frequently fails to converge within the training budget, especially at large values of p. MGPO, by contrast, converges reliably across all configurations, with the required number of updates increasing slowly with p.

D.2.1 Performance validation.

To further validate the quality of the learned policies, we compare the best MGPO run per configuration against an analytical Riccati-based baseline. For a fixed mode, the Riccati equations yield the optimal linear feedback controller; we therefore consider a baseline that selects a single mode and applies the corresponding Riccati solution. When multiple modes are available, we report the performance of the best such single-mode controller.

We begin with the case $J = 1$ for varying values of p, corresponding to the standard LQR problem, where the Riccati solution is globally optimal. This provides a ground-truth benchmark against which we can directly assess optimality. As shown in Figure 8a, MGPO achieves near-optimal performance across all dimensions. While a slight degradation is observed as p increases, it remains very modest, with relative gaps below 0.2% even at $p = 48$.

We additionally consider the S-LQR setting with $J = 4$. In this case, the Riccati baseline corresponds to the best fixed-mode controller, and thus cannot exploit switching. Figure 8b shows that MGPO



(a) Target gap: 5%.

(b) Target gap: 10%.

(c) Target gap: 20%.

(d) Target gap: 30%.

Figure 6: Median number of policy updates required to reach a target validation-performance gap in the switched LQR problem, as the continuous action dimension p varies. Each panel uses a different target gap. Blue curves correspond to MGPO and red curves to H-PPO. Solid lines use $J = 2$ modes, while dashed lines use $J = 8$ modes. Arrows indicate that the median did not converge for the given configuration.

improves upon this baseline across all dimensions, with gains ranging from about 19% to 38%. This indicates that MGPO effectively learns to select appropriate modes when beneficial.

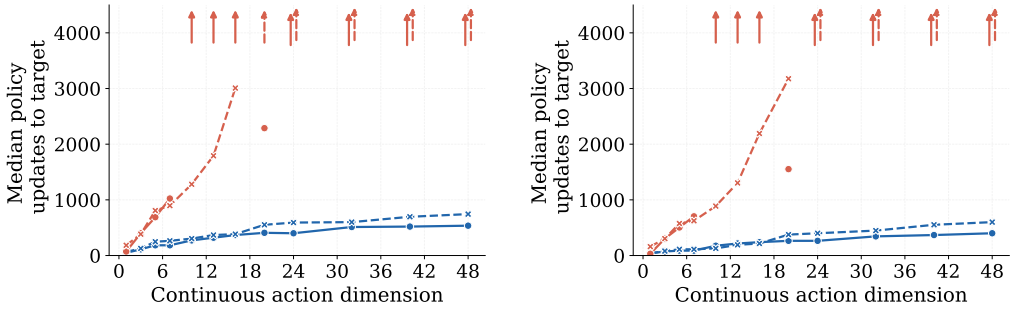
D.3 Gradient Quality Experiments

This appendix provides a complete specification of the procedure used to evaluate the quality of gradient estimators, as presented in Section 4.2.

Setting. We consider a setting with p identical products, with $p \in \{1, 3, 5, 7, 10, 20, 30, 50\}$. All products have an underage $u = 9$, holding cost $h = 1$ and i.i.d. Poisson demand with mean 10. Fixed costs are set to $K = 64p$. We focus on this simplified setting to obtain more transparent insights. In particular, assuming identical products enables meaningful comparisons across product counts, whereas heterogeneity could introduce confounding effects in our analysis.

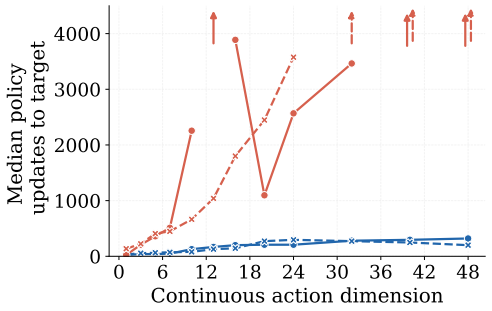
Policy construction. For each value of p, we train 3 independent policies to near-optimality using **MGPOFull**, with the hyperparameters defined in Appendix C.2. For each trained policy, we construct evaluation policies as follows. We first freeze the parameters of the continuous policy $\pi_\kappa^{\mathcal{B}}$ and modify its outputs by applying a constant shift of 2 units. This avoids PW gradients having zero magnitude, which would render alignment metrics uninformative. We additionally add i.i.d. standard Normal noise to each output so that all gradient estimators are evaluated under a common stochastic policy.

We then reinitialize the discrete policy $\pi_\phi^{\mathcal{X}}$ and train it together with a value function, while keeping the continuous component fixed. To obtain a reliable critic, we first train the value function alone for several epochs, keeping the discrete policy fixed. After this initialization phase, we jointly train the discrete policy and the value function. We use a larger learning rate for the value function so that it remains well-calibrated as the discrete policy evolves. We alternate between training epochs and

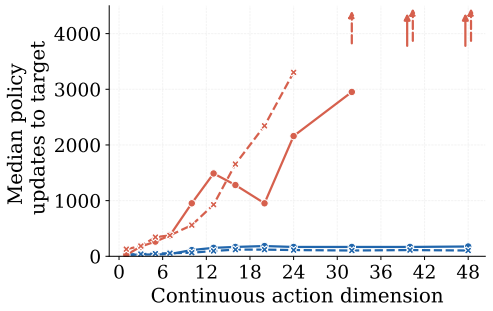


(a) Target gap: 5%.

(b) Target gap: 10%.

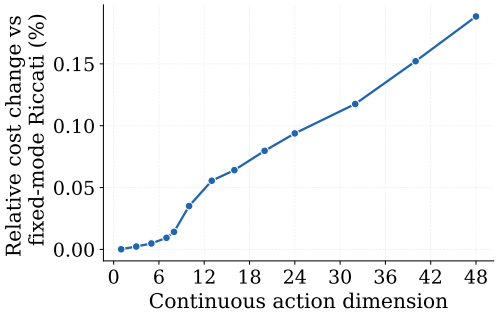


(c) Target gap: 20%.

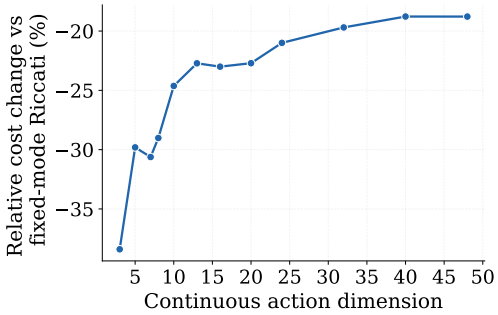


(d) Target gap: 30%.

Figure 7: Median number of policy updates required to reach a target validation-performance gap in the switched LQR problem, as the continuous action dimension p varies. Each panel uses a different target gap. Blue curves correspond to MGPO and red curves to H-PPO. Solid lines use $J = 4$ modes, while dashed lines use $J = 6$ modes. Arrows indicate that the median did not converge for the given configuration.



(a) $J = 1$ (standard LQR).



(b) $J = 4$ (switched LQR).

Figure 8: Relative cost difference between MGPO and Riccati-based baselines as a function of the continuous action dimension p. The relative cost difference is computed as the percentage change in cost achieved by MGPO compared to a Riccati baseline **(lower is better)**. For $J = 1$, the baseline is the optimal Riccati controller; for $J = 4$, the baseline is the best single-mode Riccati controller among the available modes.

gradient-estimation epochs. During gradient-estimation epochs, both policy components are frozen so that all gradient estimates correspond to the same fixed policy.

Gradient computation. We estimate the PW and cross terms using the reparameterization trick [Kingma and Welling, 2013, Rezende et al., 2014].

To estimate all SF-type terms (including the discrete gradient and the cross term), we use Monte Carlo rollouts combined with a learned critic baseline, avoiding the bias introduced by PPO-style objectives. Specifically, whenever an action-value $Q_{\pi_\theta}(s_t, a_t)$ appears in the gradient expressions (1) and (2), we replace it with the advantage $Q_{\pi_\theta}(s_t, a_t) - V_{\pi_\theta}(s_t)$.

The action-value is estimated via sampled cumulative costs, while V_{π_θ} is learned by the value network. This corresponds to a REINFORCE-style estimator with a baseline, and yields unbiased gradient estimates with reduced variance relative to vanilla REINFORCE. We refer to this gradient as SF for brevity.

D.3.1 Batch-Level Gradient Estimators

Fix a policy π_θ. A batch-level gradient estimator with a batch size of B trajectories is defined as

$$g = \frac{1}{B} \sum_{i=1}^{B} \hat{g}^{(i)}$$

where each $\hat{g}^{(i)}$ is a per-trajectory gradient estimate computed from an independent rollout.

We assume $g = \mu + \varepsilon$, where $\mu = \nabla_\theta J(\theta)$ denotes the true gradient and ε is zero-mean noise with covariance matrix Σ. In practice, the true gradient μ is not directly observable, and we approximate it using the iteration-level average mixed gradient, which serves as a low-variance proxy.

All batch estimators are constructed from disjoint sets of trajectories and are therefore conditionally independent given the policy parameters.

D.3.2 Metric Definitions

Let g and h be independent batch-level gradient estimators with true gradient μ. For each metric, we form a Monte Carlo estimate by averaging the relevant quantity over R independent draws of (g, h), and report the final metric as a function of this average as described below.

Signal. The individual-sample estimator $\langle g, h \rangle$ satisfies $\mathbb{E}[\langle g, h \rangle] = \|\mu\|^2$, providing an unbiased estimate of the squared gradient norm. We define

$$\text{Signal} = \sqrt{\hat{q}^{\text{sig},2}}, \qquad \hat{q}^{\text{sig},2} = \frac{1}{R} \sum_{r=1}^{R} \langle g^{(r)}, h^{(r)} \rangle.$$

RMSE. The individual-sample estimator $\frac{1}{2}\|g - h\|^2$ satisfies

$$\mathbb{E}\big[\tfrac{1}{2}\|g - h\|^2\big] = \mathbb{E}\big[\|g - \mu\|^2\big] = \text{Tr}(\Sigma),$$

providing an unbiased estimate of the expected squared error, equal to the trace of the covariance matrix Σ. We define

$$\text{RMSE} = \sqrt{\hat{q}^{\text{rmse},2}}, \qquad \hat{q}^{\text{rmse},2} = \frac{1}{R} \sum_{r=1}^{R} \frac{1}{2} \|g^{(r)} - h^{(r)}\|^2.$$

Alignment. The individual-sample estimator $\frac{\langle g,h \rangle}{\|g\| \, \|h\|}$ satisfies $\mathbb{E}\Big[\frac{\langle g,h \rangle}{\|g\| \, \|h\|}\Big] = \mathbb{E}[\cos(g, h)]$, measuring directional consistency across independent estimates irrespective of magnitude. We define

$$\text{Align} = \hat{q}^{\text{align}}, \qquad \hat{q}^{\text{align}} = \frac{1}{R} \sum_{r=1}^{R} \frac{\langle g^{(r)}, h^{(r)} \rangle}{\|g^{(r)}\| \, \|h^{(r)}\|}.$$

Signal-to-noise ratio. We define the SNR as the ratio of signal to RMSE:

$$\text{SNR} = \frac{\text{Signal}}{\text{RMSE}} = \sqrt{\frac{\hat{q}^{\text{sig},2}}{\hat{q}^{\text{rmse},2}}}.$$

Higher values indicate more reliable gradient estimates, as the gradient signal dominates its variability.

D.3.3 Cross-Alignment via Leave-One-Out Estimation

To evaluate how well a single-batch estimator aligns with the true gradient direction, we construct a proxy of the true gradient using the iteration-level average gradient.

Let $\{g^{(1)}, \ldots, g^{(M)}\}$ be batch-level mixed gradient estimators collected during a training iteration, and define the iteration-level mean

$$G = \frac{1}{M} \sum_{i=1}^{M} g^{(i)}.$$

To avoid self-inclusion bias, for a sampled mixed batch $g^{(j)}$ we construct the leave-one-out estimator

$$G_{-j} = \frac{1}{M-1} \sum_{i \neq j} g^{(i)}.$$

Given a test batch estimator h (corresponding to batch-level estimates of either the mixed or PW gradient in our experiments), we define

$$\text{CrossAlign}(h) = \hat{q}^{\text{calign}}, \qquad \hat{q}^{\text{calign}} = \frac{1}{R} \sum_{r=1}^{R} \frac{\langle G_{-j^{(r)}}, h^{(j^{(r)})} \rangle}{\|G_{-j^{(r)}}\| \, \|h^{(j^{(r)})}\|},$$

where $j^{(r)}$ is drawn uniformly at random from $\{1, \ldots, M\}$ in each repetition. This metric approximates the alignment between a batch-level estimator and the true gradient direction.

D.3.4 Metric estimation

At each training iteration, we collect a set Q_e of batch-level gradient estimators. Metrics are computed using a split-sample procedure.

For alignment, signal, and RMSE, we repeat the following procedure $R = 100$ times: we sample two gradients $g, h \in Q_e$ uniformly at random (with replacement) and compute the corresponding individual-sample estimator. The final estimate $\hat{q}$ is obtained by averaging over repetitions, and the reported metric is computed as a function of $\hat{q}$ as described in the previous subsection.

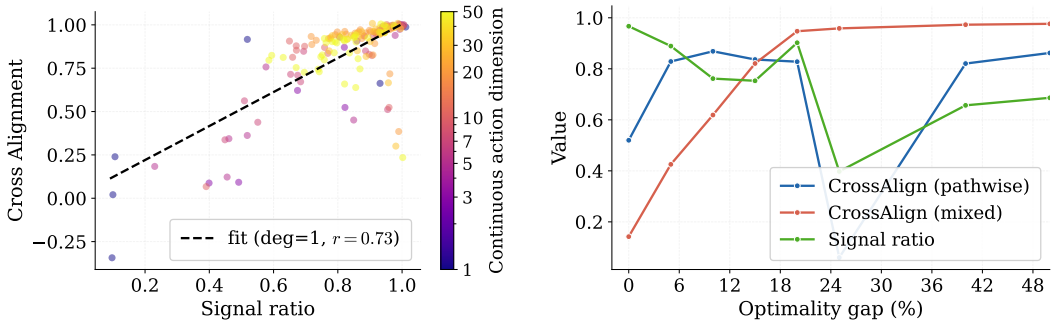
For cross-alignment, we sample an index $j^{(r)}$ uniformly at random from $\{1, \ldots, M = |Q_e|\}$ and compute the leave-one-out quantity above using the paired mixed and PW estimators corresponding to that index. This procedure is repeated $R = 100$ times and averaged.

D.3.5 Aggregation

To compare gradient quality across different training phases and problem sizes, we aggregate results according to two variables: the number of products p and the performance gap (relative to the best performance achieved after perturbing the continuous policy and retraining the discrete policy component). Concretely, for each value of p and each trained policy, we first compute the best validation loss achieved by any run of the perturbed policy. We then define the optimality gap at a given iteration as the relative difference between the current validation loss and this reference value. Performance gaps are discretized into buckets of varying width depending on the analysis. Metrics are first averaged within each iteration, then across runs, according to their performance gap bucket. Reported values correspond to averages over all runs and iterations for each $(p, \text{performance gap bucket})$ pair.

D.3.6 Results

We now present additional results to complement those presented in Section 4.2. We organize results according to key insights.



(a) CrossAlign vs. signal ratio across all (p, gap) configurations. Color indicates p (darker = smaller dimension).

(b) CrossAlign and signal ratio vs. performance gap for $p = 10$. Both series share the $[0, 1]$ range but measure different quantities: CrossAlign is a cosine similarity, while signal ratio is a relative norm.

Figure 9: Cross alignment between batch-level PW and iteration-level mixed gradient estimates across product counts and training regimes.

Pathwise-only estimator's quality is driven by signal ratio. We now provide guidance about when a PW-only estimator provides a good approximation of the mixed gradient. Our main hypothesis is that performance is governed by the *signal ratio*, defined as the fraction of total squared signal attributable to the PW component; that is, the squared signal of the PW term divided by the sum of the squared signals of the PW and cross components. Intuitively, the PW estimator performs well when most of the gradient signal is carried by the PW term.

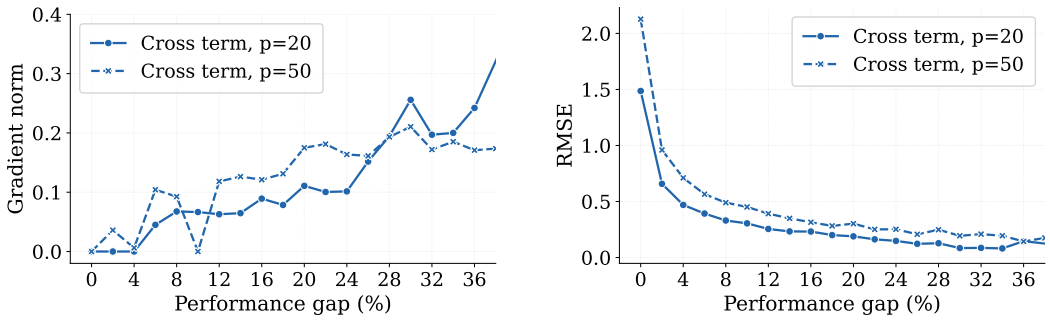
To support this, Figure 9a plots the signal ratio against its cross-alignment between batch-level PW estimates and iteration-level mixed gradients, across product counts. Each point corresponds to a $(p, \text{performance gap})$ pair, and the dashed line shows a linear fit. We observe a strong positive relationship, with $r = 0.73$, indicating that higher signal ratios are associated with better alignment with the mixed gradient. Intuitively, when the cross term has comparable or larger magnitude, omitting it significantly distorts the gradient direction.

The signal ratio varies across both problem parameters and training regimes. First, we expect larger fixed costs to reduce the signal ratio, as they increase the magnitude of the cross term (via larger advantages) relative to the PW component. Second, it depends on the discrete policy's quality. Far from optimality, the signal ratio is typically large due to very large PW gradient norms. Additionally, as the discrete policy approaches a best response, the signal ratio approaches 1 since the cross term's norm vanishes, consistent with Theorem 2.

Figure 9b illustrates the evolution of both signal ratio and cross-alignment metrics for both PW and mixed batch-level estimators, for $p = 10$. PW achieves high CrossAlign across most performance gaps, except in the 20–35% range, where a drop in signal ratio coincides with poor alignment. For gaps below 15%, PW attains higher CrossAlign than mixed, although alignment degrades overall due to increased variance in the iteration-level mixed gradient. Still, the relatively stronger batch-level alignment suggests that PW remains a better estimator in this regime. This suggests that, if PW-only estimators are to be used, the evolution of the discrete policy must be carefully controlled during training.

Cross term degrades close to optimality due to increased sensitivity of discrete policy. We analyze the drivers behind the degradation of the cross term near optimality. Figures 10a and 10b report the gradient norm and RMSE, respectively, for continuous action dimensions $p = 20$ and $p = 50$. The RMSE exhibits sharp spikes as the discrete policy approaches a best response, occurring precisely in regions where the gradient norm becomes small. This renders the gradient increasingly uninformative near optimality.

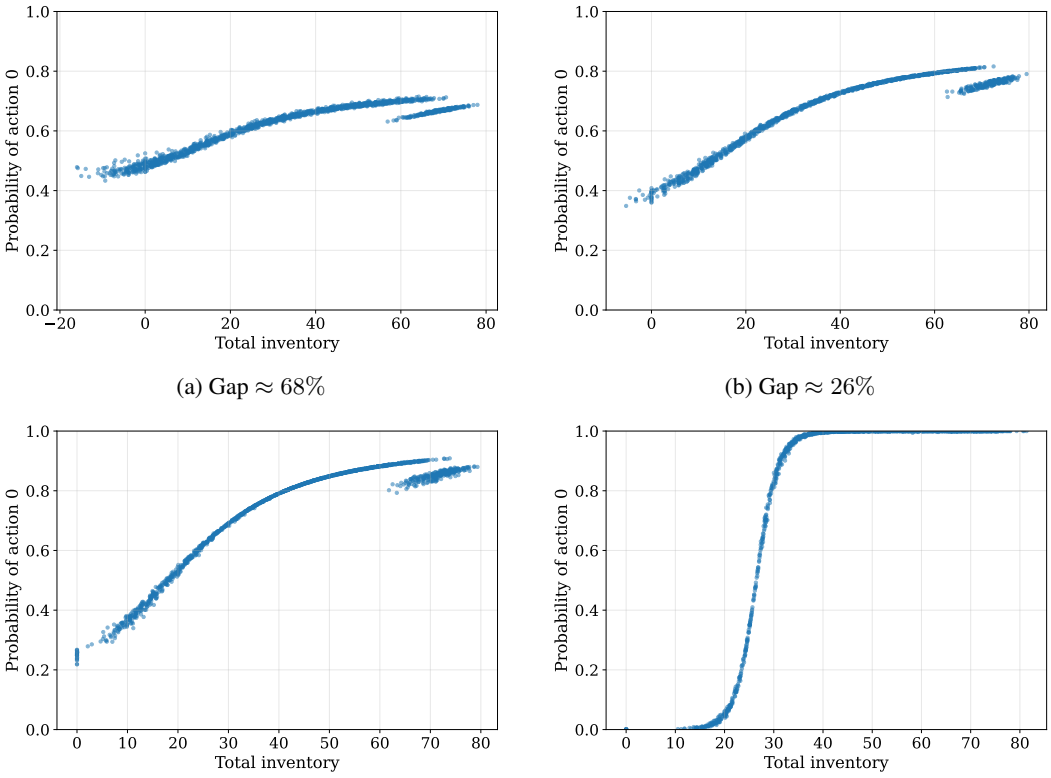
To understand this, Figure 11 plots the probability of not ordering as a function of total system on-hand and in-transit inventory across different performance gaps in the JRP with one product. As shown in Scarf [1960], under linear underage and holding costs with shared fixed costs and i.i.d. demand, the optimal policy is characterized by an (s, S) policy: there exists a number s such that



(a) Gradient norm.

(b) RMSE.

Figure 10: Gradient norm and RMSE of batch-level gradient estimates of the cross term in the JRP setting for $p = 20$ and $p = 50$.



(a) Gap $\approx 68\%$

(b) Gap $\approx 26\%$

(c) Gap $\approx 10\%$

(d) Gap $\approx 0\%$

Figure 11: Probability of not placing an order as a function of total system inventory, for several performance gaps.

if total inventory falls on or below s, we should place an order that increases total inventory to S. We observe that the learned policy seems to approximate such a thresholded structure, becoming increasingly sharp as the performance gap decreases.

This implies that the sensitivity of the discrete policy to the state becomes highly uneven around the threshold. The policy is nearly flat far from the threshold but becomes very steep in its vicinity. As a result, the SF term $\nabla_\kappa \log \pi_\phi^{\mathcal{X}}(x_t \mid s_t(\kappa))$ exhibits high variability across trajectories, depending on whether the state lies near this transition region. This variability drives the increase in the variance of the cross term, even as its expectation decreases.

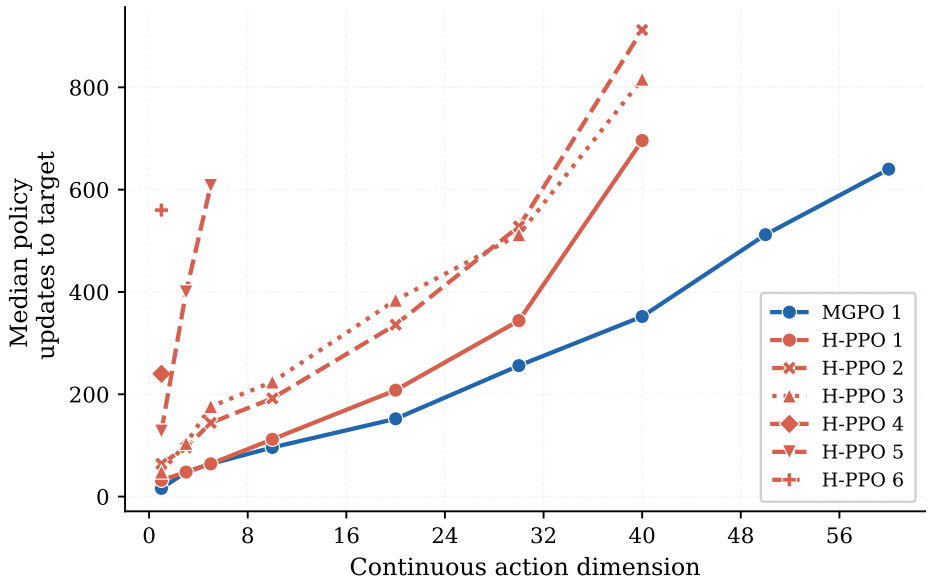


Figure 12: Median policy updates to reach a 10% target gap on the JRP for MGPO and six H-PPO hyperparameter configurations with batch size $B = 64$, as the continuous action dimension p varies. Configurations for which the median did not converge within the training budget are omitted.

D.4 H-PPO Hyperparameter Robustness

Figure 12 reports a hyperparameter ablation study for H-PPO, comparing one-off variants against the baseline H-PPO and MGPO configurations (H-PPO 1 and MGPO 1, respectively). The baseline H-PPO 1 uses learning rate 10^{-4}, entropy coefficient $\beta_{\text{ent}} = 0.5$ and 5 optimization epochs per iteration (i.e., each collected batch is reused for 5 gradient passes) in which both the discrete and continuous policies are updated. Each one-off variant isolates a single change to one of these knobs: H-PPO 2 lowers entropy regularization to $\beta_{\text{ent}} = 0.01$; H-PPO 3 increases the number of H-PPO epochs per iteration from 5 to 10; H-PPO 4 enables discrete-only extra epochs at the base learning rate 10^{-4}; H-PPO 5 increases the learning rate to 10^{-3} without extra discrete epochs; and H-PPO 6 combines both (10^{-3} with discrete-only extra epochs). The discrete-only extra epoch variants (H-PPO 4 and H-PPO 6) deserve special attention: in the base setup, 5 gradient steps are taken with respect to both discrete and continuous parameters simultaneously. Enabling discrete-only extra epochs instead performs 5 full gradient passes for the discrete parameters but only a single pass for the continuous ones. The motivation is to mirror how MGPO operates, where discrete parameters take 5 PPO epochs per batch but the continuous only take 1, thereby testing whether a comparable update structure can close the gap between H-PPO and MGPO. An additional variant with 1 optimization epoch per batch was included in the sweep but is omitted from the figure, as it failed to satisfy the convergence criterion for any value of p.

Across all variants, baseline H-PPO 1 is the strongest H-PPO configuration. All variants converge more slowly than H-PPO 1 for most values of p, with several exhibiting erratic behavior at small p before diverging steeply. Notably, even the discrete-only extra epoch variants, which were specifically designed to mimic MGPO's update structure, fail to match H-PPO 1 let alone MGPO 1. In contrast, MGPO 1 outperforms every H-PPO variant across the full range of p, with the gap widening consistently as p grows. These results suggest that MGPO's advantage over H-PPO is robust to hyperparameter tuning: even with careful one-off adjustments, no H-PPO variant approaches MGPO's convergence speed at larger action dimensions.

D.5 Comparison with HyAR

We additionally compare against HyAR [Li et al., 2021], a hybrid-action baseline that runs TD3 in a latent action space learned by a variational autoencoder, on both the JRP and S-LQR settings. HyAR is

given the same number of environment transitions as MGPO and H-PPO (2.56M for JRP with $B = 16$, and 10.24M for JRP with $B = 64$ and for S-LQR), with its checkpoint selected on the validation set within that budget. For JRP, we run HyAR for all product counts $p \in \{1, 3, 5, 10, 20, 30, 40, 50, 60\}$; for S-LQR, for $p \in \{3, 8, 16, 32, 48\}$ and $J \in \{2, 4, 6, 8\}$. To reduce computational costs, we only use 5 independent trials per configuration for HyAR. Tables 5 and 6 report the ratio of the median test loss of HyAR and H-PPO to that of **MGPOFull**, so that values above 1 indicate a higher cost than **MGPOFull**.

HyAR only performs well in JRP with few products: at $p = 1$ it is within 0.3% of **MGPOFull**, but its performance degrades quickly as p grows, reaching ratios of 2.9 and 5.5 at $p = 60$ for $B = 64$ and $B = 16$, respectively. In S-LQR, HyAR is already 20% to 64% worse than **MGPOFull** at $p = 3$, and the gap widens with p, exceeding a factor of 2.5 at $p = 48$ for all J. Overall, H-PPO outperforms HyAR in all JRP configurations and in all but three S-LQR configurations. The exceptions ($p = 32$, $J = 8$ and $p = 48$, $J \in \{6, 8\}$) are configurations where H-PPO fails to converge and its median loss is 4 to 125 times that of **MGPOFull** (Table 4). Neither baseline approaches the performance of MGPO at large p.

Table 4: S-LQR test loss across action dimensions p and modes J. Mean $\pm$ std and median over 10 independent trials.

		MGPOFull		**MGPONoCross**		**H-PPO**	
p	J	**Mean $\pm$ Std**	**Median**	**Mean $\pm$ Std**	**Median**	**Mean $\pm$ Std**	**Median**
1	$J=2$	36.83 ± 0.00	36.83	36.82 ± 0.00	36.82	36.84 ± 0.00	36.84
1	$J=4$	36.83 ± 0.00	36.83	36.82 ± 0.00	36.82	36.84 ± 0.00	36.84
1	$J=6$	37.51 ± 0.00	37.51	37.51 ± 0.00	37.51	37.52 ± 0.00	37.52
1	$J=8$	37.52 ± 0.00	37.52	37.51 ± 0.00	37.51	37.52 ± 0.00	37.52
3	$J=2$	45.26 ± 0.01	45.25	45.23 ± 0.01	45.23	45.38 ± 0.03	45.38
3	$J=4$	52.30 ± 0.03	52.31	52.21 ± 0.02	52.22	52.55 ± 0.04	52.55
3	$J=6$	50.96 ± 0.02	50.96	50.89 ± 0.02	50.88	51.17 ± 0.02	51.18
3	$J=8$	50.96 ± 0.03	50.96	50.89 ± 0.02	50.90	51.17 ± 0.03	51.17
5	$J=2$	47.03 ± 0.03	47.01	46.99 ± 0.02	46.98	47.58 ± 0.14	47.54
5	$J=4$	59.73 ± 0.05	59.74	59.47 ± 0.05	59.48	60.29 ± 0.05	60.29
5	$J=6$	63.28 ± 0.04	63.27	63.02 ± 0.05	63.03	63.95 ± 0.05	63.97
5	$J=8$	63.33 ± 0.06	63.32	63.02 ± 0.04	63.02	63.92 ± 0.06	63.93
7	$J=2$	48.28 ± 0.03	48.27	48.24 ± 0.03	48.24	53.18 ± 1.20	53.46
7	$J=4$	61.80 ± 0.04	61.79	61.36 ± 0.04	61.37	62.52 ± 0.11	62.55
7	$J=6$	68.60 ± 0.08	68.58	67.93 ± 0.04	67.91	69.29 ± 0.04	69.29
7	$J=8$	72.34 ± 0.10	72.34	71.51 ± 0.04	71.52	73.11 ± 0.09	73.12
8	$J=2$	49.03 ± 0.06	49.04	48.93 ± 0.03	48.92	54.87 ± 0.36	54.82
8	$J=4$	65.61 ± 0.08	65.63	64.97 ± 0.05	64.96	66.53 ± 0.15	66.54
8	$J=6$	69.92 ± 0.09	69.92	69.22 ± 0.04	69.21	70.79 ± 0.10	70.78
8	$J=8$	77.26 ± 0.05	77.26	76.19 ± 0.05	76.17	78.22 ± 0.08	78.22
10	$J=2$	50.85 ± 0.07	50.81	50.75 ± 0.03	50.73	57.66 ± 0.48	57.58
10	$J=4$	65.30 ± 0.07	65.32	64.92 ± 0.08	64.91	76.28 ± 4.65	77.69
10	$J=6$	72.88 ± 0.07	72.90	72.14 ± 0.07	72.14	74.05 ± 0.09	74.06
10	$J=8$	77.96 ± 0.08	78.00	76.92 ± 0.08	76.94	79.35 ± 0.16	79.28
13	$J=2$	50.97 ± 0.08	50.99	50.84 ± 0.03	50.84	58.84 ± 0.44	58.82
13	$J=4$	67.56 ± 0.07	67.56	66.98 ± 0.07	67.00	81.70 ± 6.89	84.48
13	$J=6$	76.57 ± 0.09	76.56	75.58 ± 0.06	75.58	78.38 ± 0.12	78.36
13	$J=8$	81.23 ± 0.08	81.26	80.01 ± 0.08	80.00	83.80 ± 0.44	83.62
16	$J=2$	52.17 ± 0.09	52.15	51.98 ± 0.07	52.00	58.93 ± 1.98	59.73
16	$J=4$	70.35 ± 0.04	70.35	69.64 ± 0.10	69.60	81.52 ± 7.99	81.79
16	$J=6$	80.01 ± 0.11	79.99	78.79 ± 0.04	78.80	82.85 ± 0.30	82.81
16	$J=8$	83.86 ± 0.08	83.84	82.44 ± 0.04	82.46	88.61 ± 0.80	88.45
20	$J=2$	52.78 ± 0.07	52.78	52.56 ± 0.08	52.60	55.19 ± 2.38	54.47
20	$J=4$	72.13 ± 0.15	72.13	71.37 ± 0.10	71.39	74.90 ± 0.72	74.59
20	$J=6$	80.66 ± 0.13	80.67	79.50 ± 0.08	79.49	85.86 ± 1.26	85.97
20	$J=8$	85.60 ± 0.12	85.57	84.27 ± 0.10	84.23	95.99 ± 3.57	95.24
24	$J=2$	52.64 ± 0.07	52.62	52.45 ± 0.09	52.45	54.34 ± 0.10	54.39
24	$J=4$	72.60 ± 0.16	72.52	71.86 ± 0.11	71.83	79.25 ± 1.72	79.91
24	$J=6$	83.49 ± 0.13	83.53	82.25 ± 0.12	82.26	95.96 ± 3.15	95.15
24	$J=8$	89.42 ± 0.15	89.42	88.04 ± 0.17	88.08	772.61 ± 1046.33	120.65
32	$J=2$	53.20 ± 0.10	53.21	52.85 ± 0.07	52.84	55.20 ± 0.14	55.18
32	$J=4$	73.24 ± 0.22	73.22	72.66 ± 0.16	72.64	83.08 ± 3.31	81.42
32	$J=6$	84.80 ± 0.14	84.83	83.38 ± 0.09	83.35	151.02 ± 74.75	114.81
32	$J=8$	91.43 ± 0.10	91.46	90.06 ± 0.11	90.08	2037.59 ± 2494.29	359.47
40	$J=2$	53.26 ± 0.15	53.24	52.80 ± 0.05	52.80	55.22 ± 0.12	55.22
40	$J=4$	74.91 ± 0.12	74.94	73.94 ± 0.26	73.87	97.78 ± 6.44	99.55
40	$J=6$	87.28 ± 0.24	87.29	85.72 ± 0.13	85.74	3356.44 ± 2652.95	3210.90
40	$J=8$	92.84 ± 0.16	92.85	91.24 ± 0.12	91.25	4978.72 ± 1284.16	5228.96
48	$J=2$	53.91 ± 0.20	53.88	53.28 ± 0.08	53.28	58.33 ± 1.09	57.85
48	$J=4$	76.15 ± 0.20	76.18	74.99 ± 0.24	74.87	119.01 ± 14.68	118.07
48	$J=6$	88.65 ± 0.27	88.59	86.76 ± 0.14	86.76	9760.17 ± 3735.48	9111.29
48	$J=8$	94.41 ± 0.33	94.34	92.59 ± 0.17	92.64	11324.53 ± 1546.91	11841.05

Table 5: JRP: ratio of the median test loss of HyAR and H-PPO to that of MGPOFull at equal numbers of environment transitions (2.56M for $B = 16$, 10.24M for $B = 64$). Values above 1 mean a higher cost than MGPOFull; the lower ratio in each row and budget is in bold.

	$B = 16$		$B = 64$	
p	**HyAR / MGPOFull**	**H-PPO / MGPOFull**	**HyAR / MGPOFull**	**H-PPO / MGPOFull**
1	1.003	**0.999**	1.002	**1.000**
3	1.071	**1.000**	1.041	**1.000**
5	1.149	**0.994**	1.100	**0.997**
10	1.187	**0.996**	1.138	**0.998**
20	1.684	**1.003**	1.276	**1.001**
30	2.056	**1.318**	1.657	**1.002**
40	2.909	**1.731**	2.023	**1.001**
50	4.448	**1.875**	2.080	**1.223**
60	5.503	**1.782**	2.864	**1.465**

Table 6: S-LQR: ratio of the median test loss of HyAR and H-PPO to that of MGPOFull on the configurations where HyAR was run (10.24M environment transitions). Values above 1 mean a higher cost than MGPOFull; the lower ratio in each row is in bold.

p	J	**HyAR / MGPOFull**	**H-PPO / MGPOFull**
3	$J = 2$	1.636	**1.003**
3	$J = 4$	1.316	**1.005**
3	$J = 6$	1.261	**1.004**
3	$J = 8$	1.199	**1.004**
8	$J = 2$	1.398	**1.118**
8	$J = 4$	1.479	**1.014**
8	$J = 6$	1.348	**1.012**
8	$J = 8$	1.448	**1.012**
16	$J = 2$	1.549	**1.145**
16	$J = 4$	1.600	**1.163**
16	$J = 6$	2.028	**1.035**
16	$J = 8$	2.261	**1.055**
32	$J = 2$	86.337	**1.037**
32	$J = 4$	1.707	**1.112**
32	$J = 6$	1.814	**1.353**
32	$J = 8$	**2.223**	3.930
48	$J = 2$	3.043	**1.074**
48	$J = 4$	2.531	**1.550**
48	$J = 6$	**5.533**	102.850
48	$J = 8$	**3.931**	125.519

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly state our contributions: the MGPO framework and its mixed gradient estimator for hybrid discrete–continuous action spaces, including the reformulation of piecewise-smooth problems in hybrid form; empirical results on inventory control and switched LQR benchmarks; and the result that the cross term vanishes near a discrete best response.

2. **Limitations**

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

   Answer: [Yes]

   Justification: In the paragraph "Limited scope" in Section 1 we discuss the limitations and underlying assumptions of our work. In particular, we note that our work "assumes access at training time to an exogenous simulator".

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: All assumptions are provided (see Assumptions 2 and 3) and we give full proofs in Appendix B.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We give a complete specification of our algorithms, policy architecture, hyperparameter selection and setting generation procedure in Appendix C.

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

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

   Answer: [Yes]

   Justification: Code is available at `https://github.com/MatiasAlvo/hybrid-rl`. All data is synthetically generated and the repository includes instructions to reproduce all experiments reported in the paper.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We give a complete specification of our algorithms, policy architecture, hyperparameter selection and setting generation procedure in Appendix C.

7. **Experiment statistical significance**

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

   Answer: [Yes]

Justification: For convergence speed (median policy updates to target), we report the median over 10 independent trials, which is robust to outliers and to censored observations arising when some runs do not converge within the training budget. Figures 5, 6, and 7 vary the target gap to assess robustness of these findings. For final performance, Tables in Appendix D report mean, standard deviation, and median over 10 independent trials.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Appendix C.2 specifies the GPU type (NVIDIA A40, 46GB), the number of iterations, batch sizes, and epochs per experiment. It also reports that most runs completed in under 60 minutes and all runs in under 90 minutes, though these are approximate due to concurrent execution on a shared cluster.

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that this research conforms with it in all respects.

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: We do not foresee any negative societal impacts from this work.

11. **Safeguards**

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

    Answer: [N/A]

    Justification: This paper poses no such risks. All data is synthetically generated and the released code consists of RL training scripts with no potential for misuse.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We use the CleanRL library Huang et al. [2022b] as the basis for our H-PPO baseline implementation, which is cited in the paper. CleanRL is released under the MIT license.

13. **New assets**

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

    Answer: [Yes]

    Justification: We release the implementation of MGPO alongside this submission as a repository. The repository includes training scripts, environment implementations, and instructions to reproduce all experiments reported in the paper.

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

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

Answer: [N/A]

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

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

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

    Answer: [N/A]

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: LLMs were not used as part of the core methodology. They were used only for writing, editing, and coding assistance, which does not require declaration per the checklist guidelines.