Model-Driven Policy Optimization in Differentiable Simulators via Stochastic Exploration

Yuval Aroosh[1], **Ayal Taitler**[1]

[1]Ben-Gurion University of the Negev, Israel
yuvalaro@post.bgu.ac.il, ataitler@bgu.ac.il

Abstract

Differentiable planning enables gradient-based optimization of decision-making problems by leveraging differentiable models of system dynamics. However, in highly nonlinear and hybrid discrete–continuous domains, the resulting optimization landscapes are often ill-conditioned, with flat regions and sharp transitions that hinder effective optimization. We propose Model-Driven Policy Optimization (MDPO), a framework that introduces stochastic exploration into differentiable planning by injecting noise into the action space during optimization. Leveraging access to the model, MDPO further adapts the noise magnitude based on gradient-derived sensitivity of the trajectory objective, yielding a time-dependent exploration profile. This enables improved exploration of the objective landscape and helps escape poor local optima via dynamic allocation of exploration across timesteps and iterations. Experiments on benchmark domains demonstrate that MDPO consistently outperforms deterministic differentiable planning, including both the noise-free variant of our method and available state-of-the-art implementations, as well as model-free baselines such as PPO, significantly improving solution quality across challenging nonlinear and hybrid settings. We further analyze the evolution of the adaptive noise magnitude across both time steps and optimization iterations, providing insight into how exploration is allocated during learning.

Introduction

Many real-world decision-making systems operate in environments where structured models of the dynamics are available or can be constructed. From aerospace and robotics to energy systems and logistics, control and planning pipelines routinely rely on models that encode domain knowledge, physical laws, or engineered approximations. The proven success of such systems, for example in flight control, industrial automation, and autonomous navigation, provides strong evidence for the effectiveness of analytical models in decision making.

At the same time, recent advances in model-free learning have demonstrated impressive capabilities in domains where accurate models are unavailable or difficult to specify, while also pushing forward modern optimization and learning tools, such as automatic differentiation. However, when analytical models are available, it is natural to ask how to best exploit them to improve decision quality, sample efficiency, and robustness. In particular, models provide rich structural information that can, in principle, be used to guide optimization and reasoning over long horizons, rather than relying solely on trial-and-error learning.

This perspective has recently given rise to a new paradigm for model-based decision making, often referred to as *differentiable planning* (Schulman et al. 2015). Building on advances in automatic differentiation and deep learning frameworks, this approach compiles planning problems into differentiable computation graphs, enabling gradient-based optimization over action sequences and parameterized policies. In particular, planning by backpropagation (Wu, Say, and Sanner 2017) has shown that, given a differentiable model, action sequences can be optimized directly by propagating gradients through the system dynamics. Subsequent work has extended this framework to stochastic domains via reparameterization techniques (Bueno et al. 2019) and neural polices, and more recently to hybrid discrete-continuous settings through differentiable relaxations of non-smooth dynamics (Gimelfarb, Taitler, and Sanner 2024). Interestingly, similar ideas have also been explored in differentiable simulation and model-based reinforcement learning, where gradients through learned or analytical dynamics are used for control and planning (Song, Kim, and Scaramuzza 2025; Heiden et al. 2021; Hafner et al. 2020). Together, these approaches establish a unified and scalable framework for model-based decision making via gradient-based optimization.

However, despite these advances, differentiable planning methods often struggle in highly nonlinear and hybrid domains due to the structure of the induced optimization landscape, which is often poorly conditioned and highly nonconvex (Taitler et al. 2024). Gradients can become uninformative in large flat regions or saturated regimes, while relaxations of discrete dynamics introduce additional plateaus and sharp transitions. As a result, optimization is dominated by poor local minima, slow convergence, and sensitivity to initialization and hyperparameters. Crucially, these challenges persist even when the underlying model is accurate, suggesting that the primary limitation lies not only in modeling, but in the *optimization dynamics* used to solve the resulting problems.

In this work, we introduce *Model-Driven Policy Optimization* (MDPO), a framework for optimization in differentiable

Copyright © 2026, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

simulators via stochastic exploration. To address the challenges posed by flat and poorly conditioned optimization landscapes, we incorporate controlled stochasticity into the optimization process, inspired by exploration mechanisms in model-free reinforcement learning (RL) (Sutton, Barto et al. 1998), but adapted to leverage access to the planning model. While differentiable planning leverages models to guide optimization, it typically lacks explicit mechanisms to explore the optimization landscape, in contrast to RL methods that rely on stochastic policies or action noise to encourage exploration. We inject noise into the action space, analogous to action noise and entropy-based exploration in RL, enabling the optimizer to escape flat regions and saddle points and explore more informative directions of the objective landscape. The magnitude of this noise can be fixed, scheduled, or adaptively determined based on properties of the local optimization landscape, which we estimate using the planning model, allowing for structure-aware exploration. We show that this approach significantly improves performance in challenging nonlinear and hybrid domains, without altering the underlying model or its relaxation.

Contributions. (i) We identify optimization pathologies in differentiable planning, particularly in hybrid discrete–continuous domains; (ii) we propose a stochastic action-space exploration mechanism with adaptive, structure-aware noise scaling; and (iii) we empirically demonstrate substantial gains in solution quality and convergence across the classes of hybrid nonlinear domains considered.

Related Work

Stochastic Optimization. The use of gradient-based methods in stochastic computation graphs has been formalized in prior work (Schulman et al. 2015). Two main classes of estimators are commonly used: pathwise (reparameterization-based) gradients (Kingma and Welling 2013) and score-function estimators (Williams 1992), with numerous recent works proposing improved low-variance estimators and relaxations (Tucker et al. 2017; Grathwohl et al. 2018; Yin and Zhou 2019). Differentiable planning methods predominantly rely on pathwise gradients due to their lower variance, but remain sensitive to well-known issues such as vanishing gradients (Pascanu, Mikolov, and Bengio 2013), poor conditioning (Nocedal and Wright 2006), and long-horizon credit assignment (Sutton et al. 1999). Various techniques have been proposed in the broader literature to improve optimization in such settings, including variance reduction (Greensmith, Bartlett, and Baxter 2004), gradient clipping (Pascanu, Mikolov, and Bengio 2013), and truncated backpropagation through time (wer 2002), though their role in improving optimization in differentiable planning settings remains largely underexplored.

Exploration in RL. Exploration is a central component of reinforcement learning, where stochasticity is introduced to improve coverage of the state–action space and avoid premature convergence (Sutton, Barto et al. 1998). Common approaches include stochastic policies (Williams 1992), entropy regularization (Haarnoja et al. 2018), and the injection of action noise (Lillicrap et al. 2016), particularly in continuous control settings. Recent work has further explored exploration strategies based on intrinsic motivation and uncertainty estimation (Pathak et al. 2017; Burda et al. 2019; Dulac-Arnold et al. 2021). These mechanisms are known to improve optimization by encouraging exploration of the objective landscape and helping escape poor local minima and saddle points, particularly in high-dimensional and non-convex settings. In contrast, differentiable planning methods typically operate in a deterministic, model-based optimization regime, where the model is fixed and assumed to be accurate, and thus lack explicit exploration mechanisms during optimization, despite having access to the underlying model. Our work bridges this gap by incorporating exploration-inspired stochasticity into the optimization process, enabling more effective model-driven policy optimization in differentiable simulators.

Background and Optimization Challenges in Differentiable Planning

Markov decision processes. We consider a finite-horizon Markov decision process (MDP) with state space $\mathcal{S}$, action space $\mathcal{A}$, transition model $p(s_{t+1}|s_t, a_t)$, and reward function $r(s_t, a_t)$. Starting from an initial state s_0, the objective is to maximize the expected cumulative reward over a horizon H, by finding a policy $\pi : \mathcal{S} \to \mathcal{A}$:

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{H-1} r(s_t, a_t)\right]. \tag{1}$$

The expectation is taken w.r.t. the trajectory distribution induced by $a_t = \pi(s_t)$ and $p(s_{t+1}|s_t, a_t)$.

Differentiable Planning in Continuous, Stochastic, and Hybrid Domains

Differentiable planning formulates decision making as gradient-based optimization over a differentiable model of the dynamics. In deterministic settings (Wu, Say, and Sanner 2017), the transition model can be written as

$$s_{t+1} = f(s_t, a_t), \tag{2}$$

and the objective in Eq. (1) reduces to a deterministic function of the trajectory.

Deterministic differentiable planning. Planning can be posed as optimizing the action sequence $\mathbf{a} = (a_0, \ldots, a_{H-1})$:

$$\max_{\mathbf{a}} \; J(\mathbf{a}) \quad \text{s.t.} \quad s_{t+1} = f(s_t, a_t). \tag{3}$$

Unrolling Eq. (2) yields a differentiable computation graph, enabling gradient-based optimization (Nocedal and Wright 2006) via backpropagation:

$$\nabla_{a_t} J = \sum_{k=t}^{H-1} \frac{\partial r(s_k, a_k)}{\partial s_k} \frac{ds_k}{da_t} + \frac{\partial r(s_t, a_t)}{\partial a_t}, \tag{4}$$

where sensitivities are computed recursively using the transition Jacobians:

$$\frac{ds_{k+1}}{ds_k} = \frac{\partial f}{\partial s}(s_k, a_k), \quad \frac{ds_{k+1}}{da_k} = \frac{\partial f}{\partial a}(s_k, a_k). \tag{5}$$

Policy parameterization. Instead of optimizing actions directly, we optimize a parametric policy $a_t = \pi_\theta(s_t)$, where π_θ is a differentiable function (e.g., a neural network). This formulation, often referred to as deep reactive policies (DRP) (Bueno et al. 2019), enables optimization over policy parameters via backpropagation through the system dynamics. This yields the objective

$$J(\theta) = \mathbb{E}\left[\sum_{t=0}^{H-1} r(s_t, \pi_\theta(s_t))\right].$$

In deterministic settings, this reduces to a standard differentiable program where gradients $\nabla_\theta J$ are obtained by backpropagation through both the policy and the dynamics.

Stochastic dynamics and reparameterization. In stochastic domains, transitions depend on exogenous noise:

$$s_{t+1} \sim p(\cdot \mid s_t, a_t),$$
$$\text{or equivalently}$$
$$s_{t+1} = f(s_t, a_t, \xi_t), \quad \xi_t \sim p(\xi).$$

The objective becomes

$$J(\theta) = \mathbb{E}_\xi\left[\sum_{t=0}^{H-1} r(s_t, \pi_\theta(s_t))\right].$$

Using reparameterization, the expectation is rewritten over noise variables independent of θ, allowing gradients to pass through the trajectory:

$$\nabla_\theta J(\theta) = \mathbb{E}_\xi\left[\sum_{t=0}^{H-1} \frac{dr(s_t, \pi_\theta(s_t))}{d\theta}\right].$$

This transformation converts the stochastic computation graph into a fully differentiable one, enabling low-variance gradient estimation via standard backpropagation.
In practice, expectations are approximated using Monte Carlo rollouts:

$$\hat{J}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{t=0}^{H-1} r(s_{i,t}, \pi_\theta(s_{i,t})),$$

which defines the optimization objective used in stochastic gradient descent (Bueno et al. 2019).

Hybrid discrete–continuous domains. In hybrid domains, the transition model includes discrete components and logical structure that are not directly differentiable. To enable gradient-based optimization, the system and reward are approximated by a differentiable surrogate:

$$s_{t+1} \approx \tilde{f}(s_t, a_t, \xi_t),$$

where discrete operations are replaced by smooth relaxations, such as t-norms (Hájek 2001).

For example, Boolean conjunction and conditional branching may be approximated as:

$$a \wedge b \approx T(a, b), \quad \text{if } c \text{ then } a \text{ else } b \approx ca + (1-c)b,$$

yielding a fully differentiable computation graph over relaxed variables (Gimelfarb, Taitler, and Sanner 2024). However, this induces a fundamental trade-off: faithful approximations yield ill-conditioned objectives, while stronger relaxation makes optimization easier at the cost of distorting the underlying problem.

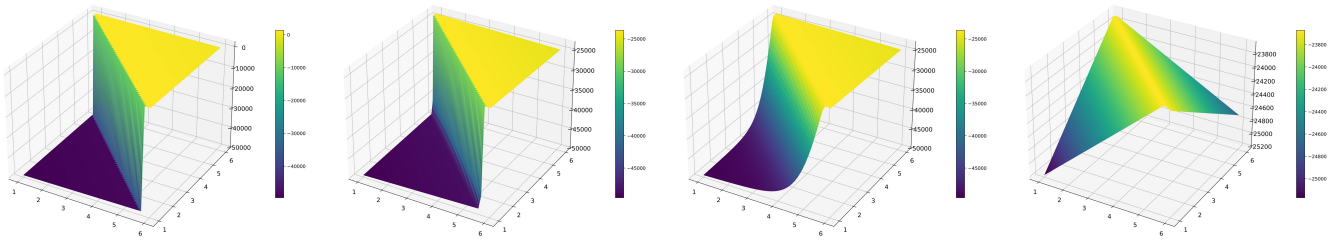
Optimization Landscape Pathologies

We now analyze the optimization landscape induced by the formulation in Section and highlight key challenges that arise in hybrid domains. In particular, we focus on the effects of differentiable relaxations on the geometry of the objective function and their implications for gradient-based optimization.

Motivating example: Power generation domain. The `Power generation` (`PowerGen`) domain is a continuous relaxation of the classical unit commitment problem presented in the 2023 International Planning Competition (IPC) (Taitler et al. 2024), where the objective is to meet demand using a set of generators while minimizing operational costs. Each generator is controlled by a continuous action representing its production level, subject to capacity constraints and cost functions that include fixed and variable components. The underlying dynamics include threshold-based and logical conditions (e.g., activation costs and feasibility constraints), which introduce non-smooth behavior in the exact formulation. In this example, we consider two generators with actions $a^{(1)}, a^{(2)} \in [1, 6]$, and evaluate constant policies over a fixed horizon, allowing us to visualize the induced objective landscape of $J(a^{(1)}, a^{(2)})$ as a function of the constant action pair.

Figure 1(a) shows the true objective surface under the exact (non-relaxed) dynamics. The landscape exhibits sharp discontinuities induced by underlying discrete decisions (e.g., activation thresholds and logical constraints), resulting in large flat regions separated by abrupt transitions. In such regions, gradients are either undefined or uninformative, rendering gradient-based optimization ineffective. To enable gradient-based optimization, hybrid dynamics are approximated using smooth relaxations (Section), controlled by a parameter w (e.g., sigmoid or t-norm sharpness). Figures 1(b)–(d) illustrate the effect of varying this parameter. For large values of w (Figure 1(b)), the relaxation closely approximates the original dynamics, preserving the overall structure of the objective. However, the resulting surface remains highly ill-conditioned, with steep transitions and extended flat regions, leading to poor gradient signal. For moderate values (Figure 1(c)), the surface becomes smoother and gradients are better behaved. However, this smoothing begins to alter the geometry of the objective, shifting optima and distorting the relative quality of solutions. For small values (Figure 1(d)), the landscape becomes fully smooth and easy to optimize, but significantly deviates from the true objective, potentially leading to solutions that are suboptimal or infeasible under the original dynamics.

Trade-off between fidelity and optimization. These observations highlight a fundamental trade-off: relaxations that closely approximate the original problem yield differentiable but poorly conditioned optimization landscapes, while more



(a) Exact: discontinuous landscape

(b) $w = 50$:sharp but differentiable

(c) $w = 2$: smoother, distorted

(d) $w = 0.01$: fully smooth, low fidelity

Figure 1: Optimization landscapes in the `PowerGen` domain under different levels of relaxation. The exact dynamics (left) produce a non-smooth objective with discontinuities and flat regions. Increasing relaxation smooths the surface but introduces a trade-off between optimization tractability and fidelity to the original problem.

aggressive smoothing improves gradient behavior at the cost of fidelity to the true objective. As a result, gradient-based optimization in differentiable planning is inherently challenging. Optimizers may become trapped in flat regions, fail to traverse sharp transitions, or converge to solutions that are optimal for the relaxed problem but suboptimal for the true dynamics. These challenges persist even when the underlying model is accurate and the relaxation is carefully tuned. This motivates the need for optimization strategies that explicitly account for the structure of the landscape and enable more effective exploration of the action space.

Model-Driven Policy Optimization via Stochastic Exploration

We introduce *Model-Driven Policy Optimization* (MDPO), a framework that augments differentiable planning with stochastic exploration in the action space. We consider a parametric policy π_θ (implemented as a neural network), and optimize its parameters via gradient-based methods. The key idea is to replace deterministic policy execution with a stochastic perturbation, enabling improved exploration of the optimization landscape while preserving end-to-end differentiability. Unlike standard stochastic optimization techniques that inject isotropic noise, MDPO introduces structured stochasticity guided by the planning dynamics and model structure. We refer to this as model-driven stochasticity because perturbations are applied at the level of action trajectories within the planning model, ensuring that exploration respects system dynamics rather than operating directly in parameter space. Specifically, perturbations are applied in a way that preserves trajectory feasibility while encouraging exploration across discontinuities in the optimization landscape.

Stochastic Action Perturbation

Given a parametric policy $\pi_\theta(s_t)$, standard differentiable planning executes actions deterministically:

$$a_t = \pi_\theta(s_t).$$

In MDPO, we instead define a stochastic policy by injecting additive noise:

$$\tilde{a}_t = \tilde{\pi}_\theta(s_t) = \pi_\theta(s_t) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \Sigma_t),$$

where Σ is the covariance matrix controlling the magnitude and structure of the perturbation.

The system then evolves according to the (possibly stochastic or relaxed) dynamics:

$$s_{t+1} = f(s_t, \tilde{a}_t, \xi_t).$$

Objective under stochastic policy. Under this formulation, the optimization objective becomes the expected cumulative reward under the perturbed policy:

$$J(\theta) = \mathbb{E}_{\epsilon,\xi}\left[\sum_{t=0}^{H-1} r(s_t, \tilde{a}_t)\right]$$

In practice, this expectation is approximated using Monte Carlo rollouts.

Gradient estimation. Using the reparameterization $\tilde{a}_t = \pi_\theta(s_t) + \epsilon_t$, where ϵ_t is independent of θ, gradients can be computed by backpropagation through the stochastic computation graph:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\epsilon,\xi}\left[\nabla_\theta \sum_{t=0}^{H-1} r(s_t, \tilde{a}_t)\right].$$

Since ϵ_t does not depend on θ, it does not contribute to the gradient, and standard automatic differentiation can be applied.

Theorem 1. *Under the reparameterized dynamics, the gradient $\nabla_\theta J(\theta)$ can be computed by backpropagation through the stochastic computation graph, and is equivalent to differentiating the cumulative reward along sampled trajectories.*

$$\nabla_\theta J(\theta) = \mathbb{E}_{\epsilon,\xi}\left[\sum_{t=0}^{H-1} \frac{\partial r}{\partial \tilde{a}_t}\frac{\partial \pi_\theta}{\partial \theta} + \sum_{t=0}^{H-1} \frac{\partial r}{\partial s_t}\frac{\partial s_t}{\partial \theta}\right]$$

A full expansion of the gradient in terms of the system Jacobians is provided in the appendix.

The stochastic policy $\tilde{\pi}_\theta$ induces exploration directly in the action space, allowing the optimizer to probe nearby regions of the objective landscape. Such exploration is particularly important in hybrid domains, where flat regions and sharp

transitions can hinder deterministic gradient-based optimization. This stochastic perturbation enables the optimizer to explore directions that would be inaccessible under deterministic gradients, particularly in flat or saturated regions of the objective landscape. As a result, the method can escape poor local optima and discover higher-quality solutions. In this context, the covariance Σ becomes a domain-dependent parameter that governs the extent and structure of exploration, and must be chosen to effectively navigate the underlying optimization landscape.

The introduction of stochastic perturbations induces a trade-off. For small values of Σ, the perturbed policy remains close to the original deterministic policy, and the resulting objective closely approximates the original optimization problem. As Σ increases, the sampling distribution deviates further from the deterministic policy, introducing bias with respect to the original objective. At the same time, the gradient estimator becomes higher variance due to increased stochasticity in the sampled trajectories. Thus, the choice of Σ plays a critical role in balancing exploration and optimization stability.

Extension to discrete actions. While the formulation above assumes continuous action spaces, the same principle can be extended to discrete settings by introducing stochastic relaxations or sampling mechanisms (e.g., via softmax or Gumbel-based perturbations), enabling analogous exploration in hybrid or discrete domains.

Adaptive Noise Scaling

Since the planning model is fully available, we can leverage it to derive informative exploration signals directly from the optimization landscape. We propose an adaptive mechanism for controlling the magnitude of action-space exploration based on the local sensitivity of the trajectory objective. This results in a time-dependent noise profile $\{\sigma_t\}_{t=0}^{H-1}$.

We propose a two-pass procedure in which each iteration consists of two rollouts. First, an *analysis rollout* is generated using the current policy without action noise. This rollout is used to estimate the sensitivity of the objective with respect to actions along the trajectory. Based on this analysis, a timestep-dependent noise profile is constructed. Then, a second *update rollout* is performed using the derived noise profile to update the policy parameters. This separates the estimation of the optimization landscape from the stochastic optimization step.

We consider stochastic dynamics of the form

$$s_{t+1} = f(s_t, a_t, \xi_t), \quad \xi_t \sim p(\xi),$$

where ξ_t represents exogenous randomness that is not controlled by the policy. A trajectory induced by a deterministic policy and a realization of $\xi_{0:H-1}$ is

$$\tau(\theta, \xi) = (s_0, a_0, s_1, a_1, \ldots, s_H, \xi_0, \ldots, \xi_{H-1}),$$

where $a_t = \pi_\theta(s_t) \in \mathbb{R}^n, \forall t \in [0, H]$. The objective is the expected cumulative reward:

$$J(\theta) = \mathbb{E}_\xi \left[\sum_{t=0}^{H-1} r(s_t, \pi_\theta(s_t)) \right].$$

Using the reparameterized dynamics, gradients can be propagated through the trajectory while treating ξ_t as fixed samples. Given an analysis trajectory $\tau(\theta, \xi)$, we compute, for each timestep t, the multi-dimensional gradient of the vector $a_t \in \mathbb{R}^n$:

$$g_t = \nabla_{a_t} \sum_{k=0}^{H-1} r(s_k, a_k) = \left(\frac{\partial J}{\partial a_t^{(1)}}, \ldots, \frac{\partial J}{\partial a_t^{(n)}} \right)$$

which corresponds to a Monte Carlo estimate of $\nabla_{a_t} J(\theta)$. We define a scalar sensitivity score:

$$s_t = \|g_t\|_2.$$

In principle, the expectation over ξ can be estimated using multiple rollouts. In practice, we use a single trajectory sample per iteration, which was found sufficient in our experiments. To obtain comparable values across the trajectory, sensitivities are normalized using a high quantile:

$$\hat{s}_t = \text{clip}\left(\frac{s_t}{q_p + \lambda}, 0, 1 \right), \quad q_p = \text{Quantile}_p(\{s_t\}),$$

where λ is a small regularization term to avoid division by zero, and $p = 0.95$ in this work. The normalized sensitivity is mapped to a timestep-dependent noise scale:

$$\sigma_t = \sigma_{\max} + (\sigma_{\min} - \sigma_{\max})(1 - \hat{s}_t)^\alpha,$$

where $\sigma_{\min}$, $\sigma_{\max}$, and α control the range and shape of the mapping. The resulting profile is used in the update rollout:

$$\tilde{a}_t = \pi_\theta(s_t) + \eta_t, \quad \eta_t \sim \mathcal{N}(0, \sigma_t^2 I),$$

and the policy parameters are updated using the resulting trajectory. This mapping assigns larger noise to timesteps with higher sensitivity, encouraging exploration where actions have greater impact on the trajectory objective.

Experiments

Benchmark Problems We evaluate the proposed method on three benchmark domains from the probabilistic planning and RL track of 2023 IPC (Taitler et al. 2024): `PowerGen` (Padhy 2004), `Reservoir Control` (Yeh 1985), and `HVAC` (Tiğrek, Dasgupta, and Smith 2002). These domains exhibit nonlinear and hybrid discrete–continuous dynamics, making them suitable testbeds for differentiable planning. For each domain, we consider two instances, resulting in six evaluation settings. A detailed description is provided in the appendix.

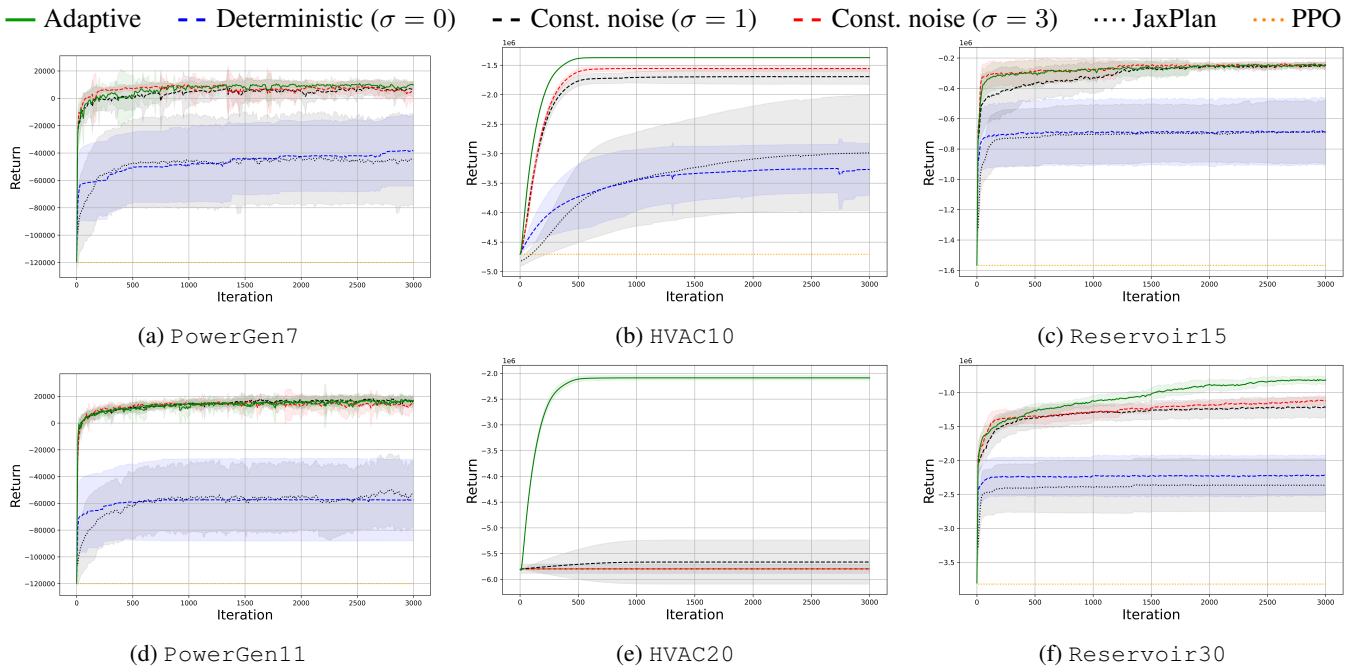
`PowerGen`. A continuous relaxation of the unit commitment problem, where generators must meet stochastic demand while minimizing production and switching costs. The domain combines continuous control with threshold-based activation and nonlinear demand. We consider instances with 7 and 11 generators.

`HVAC`. A nonlinear control problem over interconnected zones, where actions regulate airflow and heating to maintain comfort while minimizing energy cost. The dynamics involve nonlinear heat transfer and coupling between zones. We consider instances with 10 and 20 zones.

`Reservoir Control`. A network of reservoirs with

Method	HVAC		PowerGen		Reservoir	
	10	20	7	11	15	30
PPO	-4,706,559.94 ± 0.38	-5,799,263.08 ± 0.21	-119,950.01 ± 217.92	-120,000.00 ± 0.00	-1,567,404.88 ± 258.54	-3,821,075.16 ± 226.01
Deterministic	-3,268,060.37 ± 441,267.23	-5,799,274.81 ± 44.65	-38,565.41 ± 25,582.69	-57,528.37 ± 30,444.41	-683,721.06 ± 221,923.57	-2,219,640.67 ± 296,596.36
JaxPlan	-2,987,355.11 ± 979,087.04	-5,793,011.93 ± 89,429.51	-44,747.36 ± 33,219.89	-54,034.04 ± 24,745.46	-688,683.95 ± 205,670.07	-2,364,420.45 ± 388,492.15
Constant 1	-1,693,569.37 ± 129,419.57	-5,799,302.46 ± 17.11	6,753.42 ± 5,279.33	16,345.46 ± 3,977.71	-249,675.61 ± 20,927.82	-1,217,795.17 ± 151,823.21
Constant 3	-1,555,612.12 ± 47,196.07	-5,799,423.99 ± 58.96	4,622.72 ± 7,027.73	13,772.41 ± 5,851.97	-248,132.79 ± 21,170.60	-1,115,865.90 ± 69,979.64
Adaptive	**-1,371,468 ± 194.35**	**-2,086,875.93 ± 42,832.24**	**10,076 ± 3,925.23**	**16,890.15 ± 3,138.48**	**-246,613.49 ± 18,382.48**	**-815,510.27 ± 59,029.69**

Table 1: **Final performance across benchmark domains and instances.** Results are averaged over 20 seeds and 20 evaluation rollouts per seed on the true dynamics, using the mean performance over the last 5 optimization iterations. Adaptive noise consistently outperforms deterministic planning, JaxPlan, and constant-noise variants, while PPO performs poorly.



(a) PowerGen7 (b) HVAC10 (c) Reservoir15

(d) PowerGen11 (e) HVAC20 (f) Reservoir30

Figure 2: **Learning curves across benchmark instances.** Performance over 3000 optimization iterations for all methods, averaged over 20 random seeds (shaded regions indicate standard deviation). Deterministic differentiable planning and JaxPlan quickly plateau, while PPO shows limited improvement under the given optimization budget. Introducing stochastic exploration improves performance, with constant noise yielding moderate gains. The proposed adaptive method consistently achieves faster convergence and higher final performance, with the advantage becoming more pronounced in harder instances.

stochastic inflow, evaporation, and flow constraints. Actions control water release, with penalties for deviations from desired levels. We consider instances with 15 and 30 reservoirs.

Methods and Baselines. We compare MDPO against several baselines to isolate the effect of stochastic exploration.

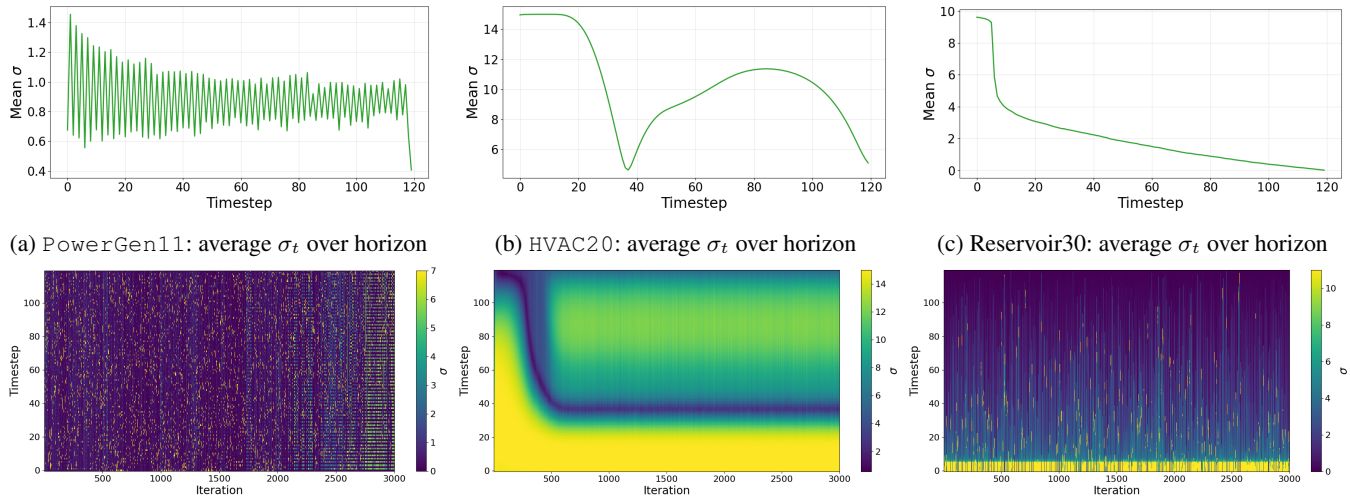
Deterministic differentiable planning. A noise-free version of our method ($\Sigma = 0$), sharing the same model, policy parameterization, and optimizer. This baseline isolates the effect of stochastic exploration.

JaxPlan. A publicly available implementation of differentiable planning based on relaxed hybrid dynamics. This serves as a strong optimization baseline representative of prior work (Gimelfarb, Taitler, and Sanner 2024).

MDPO (Constant noise). We consider fixed Gaussian noise with $\sigma \in \{1, 3\}$ applied to the action space during optimization. These variants evaluate the effect of non-adaptive stochastic exploration. These values were chosen as representative moderate noise levels that provide consistent improvements while avoiding instability, as larger noise magnitudes can lead to distributional drift and degrade optimization.

MDPO (Adaptive). Our method uses a time-dependent noise profile derived from gradient-based sensitivity. In all experiments, we set $\sigma_{\min} = 0$ and a large $\sigma_{\max}$ (typically in the range $[9, 11]$), providing a wide exploration range and allowing the method to flexibly adapt noise levels to the local optimization landscape. Results were not highly sensitive to the exact choice of $\sigma_{\max}$, provided it remained sufficiently large.

Model-free RL. We include Proximal Policy Optimization (PPO) (Schulman et al. 2017) as a model-free baseline. PPO



(a) PowerGen11: average σ_t over horizon (b) HVAC20: average σ_t over horizon (c) Reservoir30: average σ_t over horizon

(d) PowerGen11: σ_t across timesteps and iterations (e) HVAC20: σ_t across timesteps and iterations (f) Reservoir30: σ_t across timesteps and iterations

Figure 3: **Adaptive noise behavior across timesteps and optimization iterations.** Top row: average noise magnitude σ_t as a function of timestep, averaged over optimization iterations. Bottom row: heatmaps showing noise magnitude across timesteps (vertical axis) and optimization iterations (horizontal axis). Results are shown for the most challenging instance of each domain. The adaptive method produces a structured, time-dependent noise profile that varies across both timesteps and training iterations. The allocation of noise is highly dynamic and problem-dependent, with different domains exhibiting distinct patterns.

uses the same policy architecture, horizon, and initialization, and performs 10 updates per rollout.

Implementation Details. All differentiable planning methods operate on models using t-norm relaxations with weight $w = 100$, providing a close approximation to the original dynamics. MDPO was implemented in PyTorch. PPO, as a model-free baseline, is trained directly on the true (non-relaxed) environment. Policies are parameterized by feedforward neural networks with two hidden layers of 12 neurons each, ensuring that performance differences reflect optimization rather than representational capacity. All methods are trained for 3000 iterations using RMSProp (Hinton 2012) (learning rate 0.01) with horizon $H = 120$, where each iteration consists of a single rollout and gradient update. All methods share the same initialization, training budget, and optimization settings. Experiments are conducted over 20 seeds, with evaluations averaged over 20 rollouts on the true (non-relaxed) dynamics in an external pyRDDLGym (Taitler et al. 2022) implementation. All experiments were conducted on a compute cluster, with each run allocated 2 CPUs and 8GB of memory.

Overall Performance. Table 1 summarizes final performance across all benchmark instances. PPO performs poorly in all domains, showing little improvement within the given budget, highlighting the difficulty of learning without a model. Deterministic differentiable planning and JaxPlan achieve similar results, consistently converging to suboptimal solutions. Introducing stochasticity via fixed action noise improves performance, but remains sensitive to the choice of scale and varies across domains. In contrast, the adaptive noise method consistently achieves the best performance across all instances. In addition, MDPO exhibits low variance across seeds while maintaining high solution quality. PPO shows low variance only due to its lack of improvement, resulting in nearly constant returns. In HVAC20, although MDPO does not have the lowest variance, it is the only method that achieves meaningful optimization, while others remain significantly worse. These results empirically support the hypothesis that stochastic exploration mitigates optimization pathologies arising from flat and ill-conditioned regions of the objective landscape.

Learning Dynamics. Figure 2 shows learning curves across all benchmark instances. PPO remains near its initial performance throughout training. Deterministic differentiable planning and JaxPlan exhibit rapid early progress but quickly plateau, indicating that optimization becomes trapped in poor regions of the objective landscape. Adding stochasticity enables continued improvement. Fixed-noise variants often achieve faster initial gains but may stagnate depending on the noise scale. In contrast, adaptive noise maintains sustained improvement, enabling both faster convergence and continued progress over time. These effects are more pronounced in harder instances. In HVAC20 (Figure 2e) and Reservoir30 (Figure 2f), adaptive noise continues to improve while other methods plateau early. PowerGen11 presents a different regime, where relatively small noise is sufficient to find high-quality policies; the adaptive method matches this performance by adjusting the noise level appropriately, as further analyzed below.

Adaptive Noise Analysis. To better understand the behavior of the adaptive mechanism, we analyze how the noise magnitude evolves across both timesteps and optimization

iterations. We focus on the most challenging instance from each domain. Figures 3a-3c show the average noise magnitude as a function of the timestep within the planning horizon. The results reveal that noise allocation is domain-dependent and reflects the underlying dynamics. In `Reservoir30` (Figure 3c), higher noise is concentrated in earlier timesteps, indicating the importance of early decisions. In `HVAC20` (Figure 3b), noise is high at early timesteps, decreases sharply over the first third of the horizon, and then follows a non-monotonic variation over the remaining timesteps, suggesting that later decisions intermittently regain importance. In `PowerGen11` (Figure 3a), the noise magnitude stabilizes around $\sigma \approx 1$ across the horizon, consistent with the strong performance of fixed noise at this scale (Figure 2d). Figure 3 provides a joint view of how noise is allocated across both timesteps and optimization iterations. Initially, high noise is applied broadly across the trajectory. As optimization progresses, the noise distribution evolves, often becoming more concentrated on subsets of timesteps. This behavior reflects the changing sensitivity of the objective and highlights the dynamic nature of the adaptive mechanism. These results demonstrate that the adaptive method can automatically capture the structure of the optimization problem, allocating exploration to regions where it appears most beneficial both temporally within the trajectory and across the course of learning.

Conclusion and Discussion

The empirical results confirm that differentiable planning is fundamentally limited by optimization challenges arising from flat gradients and sharp transitions, particularly in nonlinear and hybrid domains. By introducing stochastic perturbations in the action space, MDPO mitigates these issues, enabling the optimizer to escape poor local optima and achieve substantially improved performance. The effectiveness of adaptive noise further highlights the importance of tailoring exploration to the local structure of the optimization landscape. This work demonstrates that the primary limitation of differentiable planning in complex domains lies not only in modeling, but in the optimization dynamics used to solve the resulting problems. Rather than modifying the model or its relaxation, MDPO improves performance by enhancing the optimization process itself through structured, model-driven exploration. This provides a simple and general approach for addressing optimization pathologies in model-based planning. More broadly, our findings highlight the role of stochastic exploration in differentiable planning. While exploration in this setting differs from its role in reinforcement learning, the parallels suggest a deeper connection between model-based gradient optimization and exploration-driven learning. Bridging these perspectives may offer promising directions for developing more robust and scalable decision-making methods.

Gradient Derivation for Stochastic Action Perturbations

In this appendix, we derive the gradient expression stated in Theorem 1.

We consider a stochastic policy obtained by perturbing a deterministic policy with additive noise:

$$\tilde{a}_t = \pi_\theta(s_t) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \Sigma_t),$$

where the noise is independent of the policy parameters θ. The system evolves according to:

$$s_{t+1} = f(s_t, \tilde{a}_t).$$

The objective is defined as:

$$J(\theta) = \mathbb{E}_\epsilon \left[\sum_{t=0}^{H-1} r(s_t, \tilde{a}_t) \right].$$

Under standard regularity conditions (e.g., smoothness and boundedness), we interchange gradient and expectation:

$$\nabla_\theta J(\theta) = \mathbb{E}_\epsilon \left[\sum_{t=0}^{H-1} \nabla_\theta r(s_t, \tilde{a}_t) \right].$$

Applying the chain rule, each term decomposes as:

$$\nabla_\theta r(s_t, \tilde{a}_t) = \frac{\partial r}{\partial s_t} \frac{ds_t}{d\theta} + \frac{\partial r}{\partial \tilde{a}_t} \frac{d\tilde{a}_t}{d\theta},$$

where $\frac{\partial}{\cdot}$ denotes partial derivatives and $\frac{d}{d\theta}$ denotes total derivatives through the computational graph.

Since ϵ_t is independent of θ, we have:

$$\frac{d\tilde{a}_t}{d\theta} = \frac{d\pi_\theta(s_t)}{d\theta} = \frac{\partial \pi_\theta(s_t)}{\partial \theta} + \frac{\partial \pi_\theta(s_t)}{\partial s_t} \frac{ds_t}{d\theta}.$$

Differentiating the system dynamics yields:

$$\frac{ds_{t+1}}{d\theta} = \frac{\partial f}{\partial s_t} \frac{ds_t}{d\theta} + \frac{\partial f}{\partial \tilde{a}_t} \frac{d\tilde{a}_t}{d\theta}.$$

Substituting the expression for $\frac{d\tilde{a}_t}{d\theta}$, we obtain:

$$\frac{ds_{t+1}}{d\theta} = \left(\frac{\partial f}{\partial s_t} + \frac{\partial f}{\partial \tilde{a}_t} \frac{\partial \pi_\theta}{\partial s_t} \right) \frac{ds_t}{d\theta} + \frac{\partial f}{\partial \tilde{a}_t} \frac{\partial \pi_\theta}{\partial \theta}.$$

Assuming that the initial state is independent of θ, we have $\frac{ds_0}{d\theta} = 0$. Unrolling the recursion gives:

$$\frac{ds_t}{d\theta} = \sum_{\tau=0}^{t-1} \left(\prod_{j=\tau+1}^{t-1} \left(\frac{\partial f}{\partial s_j} + \frac{\partial f}{\partial \tilde{a}_j} \frac{\partial \pi_\theta}{\partial s_j} \right) \right) \frac{\partial f}{\partial \tilde{a}_\tau} \frac{\partial \pi_\theta}{\partial \theta},$$

with the convention that an empty product equals the identity matrix.

Connection to Theorem 1. Substituting this expression into the gradient expansion, we obtain:

$$\nabla_\theta J(\theta) = \mathbb{E}_\epsilon \left[\sum_{t=0}^{H-1} \left(\frac{\partial r}{\partial s_t} \frac{ds_t}{d\theta} + \frac{\partial r}{\partial \tilde{a}_t} \frac{d\tilde{a}_t}{d\theta} \right) \right].$$

The first term captures the *indirect effect* of θ through the state trajectory:

$$\frac{\partial r}{\partial s_t} \frac{ds_t}{d\theta},$$

which, after substitution, corresponds to the accumulation of sensitivity through the dynamics.

The second term captures the *direct effect* through the action:

$$\frac{\partial r}{\partial \tilde{a}_t}\frac{d\tilde{a}_t}{d\theta} = \frac{\partial r}{\partial \tilde{a}_t}\left(\frac{\partial \pi_\theta}{\partial \theta} + \frac{\partial \pi_\theta}{\partial s_t}\frac{ds_t}{d\theta}\right).$$

Combining these contributions yields the expression stated in Theorem 1, consisting of: (i) a direct term corresponding to the immediate influence of actions on rewards, and (ii) an indirect term capturing the propagation of parameter influence through the system dynamics.

This establishes that the gradient can be computed via standard backpropagation through the stochastic computation graph under reparameterized action noise.

Noise Extension to Multiple Action Types

In domains such as `HVAC`, the action at time t is not a single vector but a collection of action tensors. For example, one may have

$$a_t = \left(a_t^{(1)}, a_t^{(2)}, \ldots, a_t^{(M)}\right),$$

where each $a_t^{(m)}$, for $m = 1, \ldots, M$, corresponds to a different action fluent type. In `HVAC`, these are the action tensors associated with `fan-in(zone)` and `heat-input(heater)`. which can be viewed as a structured matrix (or collection of tensors), where each block $a_t^{(m)}$ corresponds to a different action fluent type.

The trajectory objective remains a scalar,

$$J(\tau),$$

but we now differentiate it separately with respect to each action tensor:

$$g_t^{(m)} = \nabla_{a_t^{(m)}} J(\tau), \qquad m = 1, \ldots, M.$$

Each such gradient is converted into a scalar action-level sensitivity:

$$c_t^{(m)} = \left\| g_t^{(m)} \right\|_2.$$

These action-level sensitivities are then aggregated into a single timestep score. In the current implementation, the default aggregation is the mean:

$$s_t = \frac{1}{M}\sum_{m=1}^{M} c_t^{(m)} = \frac{1}{M}\sum_{m=1}^{M} \left\| \nabla_{a_t^{(m)}} J(\tau) \right\|_2.$$

Once s_t is obtained, the rest of the pipeline is unchanged: the timestep scores are normalized across the trajectory, mapped to a scalar noise level σ_t, and this same σ_t is then broadcast to all action tensors at timestep t. Thus, in the multi-action setting, the current method still produces one scalar noise level per timestep, not one separate noise level per action type.

Benchmark Domains

Power Generation Domain

The Power Generation (PowerGen) domain is a continuous relaxation of the classical unit commitment problem, where a set of generators must meet a stochastic demand while minimizing operational costs. The system evolves over a finite horizon H, with dynamics defined over continuous and discrete state variables.

State and action space. Let $\mathcal{P}$ denote the set of generators. For each generator $p \in \mathcal{P}$, the system maintains:

- a continuous production level from the previous timestep, $x_t^{(p)} \in \mathbb{R}_{\geq 0}$,
- a binary activation state $o_t^{(p)} \in \{0, 1\}$.

Additionally, the system includes a continuous exogenous temperature variable $T_t \in \mathbb{R}$.

At each timestep, the controller selects a continuous action:

$$a_t^{(p)} \in [0, a_{\max}^{(p)}],$$

representing the intended production level of generator p.

Activation and effective production. A generator is active if its production exceeds a minimum threshold:

$$o_t^{(p)} = \mathbb{I}\left[a_t^{(p)} \geq a_{\min}^{(p)}\right].$$

The effective production is defined as:

$$\tilde{a}_t^{(p)} = \begin{cases} \text{clip}\left(a_t^{(p)} + \eta_t^{(p)},\ a_{\min}^{(p)},\ a_{\max}^{(p)}\right), & \text{if } o_t^{(p)} = 1, \\ 0, & \text{otherwise}, \end{cases}$$

where $\eta_t^{(p)} \sim \text{Weibull}(k_p, \lambda_p)$ models stochastic fluctuations in production.

State transitions. The production and activation states evolve as:

$$x_{t+1}^{(p)} = \tilde{a}_t^{(p)}, \quad o_{t+1}^{(p)} = o_t^{(p)}.$$

The temperature evolves stochastically:

$$T_{t+1} = \text{clip}\big(T_t + \xi_t,\ T_{\min},\ T_{\max}\big), \quad \xi_t \sim \mathcal{N}(0, \sigma_T^2).$$

Demand model. Demand is a nonlinear function of temperature:

$$D_t = D_0 + \alpha(T_t - T^*)^2,$$

where T^* is the reference temperature minimizing demand.

The fulfilled demand is given by:

$$F_t = \min\left(D_t, \sum_{p \in \mathcal{P}} \tilde{a}_t^{(p)}\right).$$

Reward function. The reward at each timestep combines multiple components:

$$\begin{aligned} R_t = &-\sum_{p \in \mathcal{P}} c_p \tilde{a}_t^{(p)} + \rho F_t - \mathbb{I}[D_t > F_t] \cdot P \\ &- \sum_{p \in \mathcal{P}} \lambda_p \left| \tilde{a}_t^{(p)} - x_t^{(p)} \right| \\ &- \sum_{p \in \mathcal{P}} \kappa_p \mathbb{I}\left[(1 - o_t^{(p)}) \cdot o_{t+1}^{(p)}\right] \end{aligned}$$

where:

- c_p is the production cost per unit,
- ρ is the revenue per unit of fulfilled demand,
- P is the penalty for unmet demand,
- λ_p penalizes changes in production,
- κ_p penalizes switching a generator on.

HVAC Domain

The `HVAC` domain models the control of temperature in a network of interconnected zones subject to energy costs and comfort constraints. The system evolves over a finite horizon H, with nonlinear dynamics and stochastic occupancy.

State and action space. Let $\mathcal{Z}$ denote the set of zones and $\mathcal{H}$ the set of heaters. The state consists of:

- zone temperatures $\delta_t^{(z)} \in \mathbb{R}$ for each $z \in \mathcal{Z}$,
- heater temperatures $\theta_t^{(h)} \in \mathbb{R}$ for each $h \in \mathcal{H}$,
- occupancy variables $o_t^{(z)} \in \{0, 1\}$ indicating whether zone z is occupied.

The control actions are continuous:

- airflow into each zone $a_t^{(z)} \geq a_{\min}$,
- heat input to each heater $u_t^{(h)} \in \mathbb{R}$.

Occupancy dynamics. Occupancy evolves stochastically according to:

$$o_{t+1}^{(z)} = o_t^{(z)} \cdot \xi_t^{(z)}, \quad \xi_t^{(z)} \sim \text{Bernoulli}(1 - p^{(z)}),$$

where $p^{(z)}$ is the probability of becoming unoccupied.

Heater dynamics. Each heater aggregates incoming air from connected zones. Let $\mathcal{Z}(h)$ denote the zones connected to heater h. The heater temperature evolves as:

$$\begin{aligned}\theta_{t+1}^{(h)} = &- \kappa_h(\theta_t^{(h)})^2 + \theta_t^{(h)} + \frac{\Delta t}{V_h}\bar{a}_t^{(h)}\left(\bar{\delta}_t^{(h)} - T_{\text{out}}\right) \\ &+ \frac{\Delta t}{KV_h}\tilde{u}_t^{(h)},\end{aligned}$$

where:

- $\bar{a}_t^{(h)}$ is the average airflow into heater h,
- $\bar{\delta}_t^{(h)}$ is the average temperature of connected zones,
- $\tilde{u}_t^{(h)}$ is a clipped version of the heat input,
- κ_h models heat dissipation.

Zone dynamics. Zone temperatures evolve according to nonlinear heat transfer:

$$\begin{aligned}\delta_{t+1}^{(z)} = &- \kappa_z(\delta_t^{(z)} - T_{\text{out}})^2 + \delta_t^{(z)} \\ &+ \frac{\Delta t}{V_z}a_t^{(z)} \sum_{h \in \mathcal{H}(z)} \frac{\theta_t^{(h)} - \delta_t^{(z)}}{|\mathcal{H}(z)|} \\ &+ \Delta t \sum_{z' \in \mathcal{Z}} \gamma_{z,z'}(\delta_t^{(z')} - \delta_t^{(z)}),\end{aligned}$$

where:

- $\mathcal{H}(z)$ denotes heaters connected to zone z,
- $\gamma_{z,z'}$ represents thermal conductivity between adjacent zones,
- V_z is the zone volume.

Reward function. The reward penalizes energy consumption and temperature deviations from a comfort range $[\delta_{\min}^{(z)}, \delta_{\max}^{(z)}]$:

$$\begin{aligned}R_t = &- \sum_{h \in \mathcal{H}} c_h\,(\tilde{u}_t^{(h)})^2 - c_a \sum_{z \in \mathcal{Z}} (a_t^{(z)})^2 \\ &- \lambda \sum_{z \in \mathcal{Z}} \mathbb{I}[o_t^{(z)} = 1] \cdot \phi(\delta_t^{(z)}),\end{aligned}$$

where the discomfort penalty is:

$$\phi(\delta) = \begin{cases}(\delta - \delta_{\min})^2, & \delta < \delta_{\min}, \\ (\delta - \delta_{\max})^2, & \delta > \delta_{\max}, \\ 0, & \text{otherwise.}\end{cases}$$

Reservoir Control Domain

The `Reservoir` Control domain models the regulation of water levels in a network of interconnected reservoirs subject to stochastic inflow, evaporation, and physical constraints. The system evolves over a finite horizon H.

State and action space. Let $\mathcal{R}$ denote the set of reservoirs. The state consists of continuous water levels

$$\ell_t^{(r)} \in [0, \ell_{\max}^{(r)}], \quad r \in \mathcal{R}.$$

At each timestep, the controller selects a continuous action

$$a_t^{(r)} \in [0, \ell_{\max}^{(r)}],$$

representing the amount of water released from reservoir r.

Stochastic inflow and evaporation. Each reservoir receives stochastic rainfall:

$$r_t^{(r)} = |\xi_t^{(r)}|, \quad \xi_t^{(r)} \sim \mathcal{N}(0, \sigma_r^2),$$

and loses water due to evaporation:

$$e_t^{(r)} = \beta \cdot \frac{\ell_t^{(r)}}{\ell_{\text{top}}^{(r)}},$$

where β is the evaporation factor.

Flow and constraints. The effective release is clipped to feasible bounds:

$$\tilde{a}_t^{(r)} = \min\left(\ell_t^{(r)}, \max(0, a_t^{(r)})\right).$$

Water is distributed across downstream reservoirs according to the connectivity graph. Let $\mathcal{D}(r)$ denote the set of reservoirs downstream of r. The flow from r to each downstream reservoir is:

$$f_t^{(r)} = \frac{\tilde{a}_t^{(r)}}{|\mathcal{D}(r)| + \mathbb{I}[\text{sea}(r)]}.$$

The total inflow to reservoir r is:

$$i_t^{(r)} = \sum_{r' \in \mathcal{U}(r)} f_t^{(r')},$$

where $\mathcal{U}(r)$ denotes upstream reservoirs.

Overflow is defined as:

$$o_t^{(r)} = \max\left(0, \ell_t^{(r)} - \tilde{a}_t^{(r)} - \ell_{\text{top}}^{(r)}\right).$$

State transition. The reservoir level evolves as:

$$\ell_{t+1}^{(r)} = \min\left\{\ell_{\text{top}}^{(r)}, \max\left(0, x_t^{(r)}\right)\right\},$$
$$x_t^{(r)} = \ell_t^{(r)} + i_t^{(r)} + r_t^{(r)} - e_t^{(r)} - \tilde{a}_t^{(r)} - o_t^{(r)}.$$

Reward function. The objective is to maintain reservoir levels within a desired range $[\ell_{\min}^{(r)}, \ell_{\max}^{(r)}]$. The reward penalizes deviations using piecewise linear costs:

$$R_t = \sum_{r \in \mathcal{R}} \psi(\ell_{t+1}^{(r)}),$$

where

$$\psi(\ell) = \begin{cases} c_{\text{low}}^{(r)}(\ell_{\min}^{(r)} - \ell), & \ell < \ell_{\min}^{(r)}, \\ c_{\text{high}}^{(r)}(\ell - \ell_{\max}^{(r)}), & \ell_{\max}^{(r)} < \ell \leq \ell_{\text{top}}^{(r)}, \\ c_{\text{high}}^{(r)}(\ell - \ell_{\max}^{(r)}) + c_{\text{ov}}^{(r)}, & \ell > \ell_{\text{top}}^{(r)}, \\ 0, & \text{otherwise.} \end{cases}$$

References

2002. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10): 1550–1560.

Bueno, T. P.; de Barros, L. N.; Mauá, D. D.; and Sanner, S. 2019. Deep reactive policies for planning in stochastic nonlinear domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7530–7537.

Burda, Y.; Edwards, H.; Pathak, D.; Storkey, A.; Darrell, T.; and Efros, A. A. 2019. Large-scale study of curiosity-driven learning. In *ICLR*.

Dulac-Arnold, G.; Levine, N.; Mankowitz, D. J.; Li, J.; Paduraru, C.; Gowal, S.; and Hester, T. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110(9): 2419–2468.

Gimelfarb, M.; Taitler, A.; and Sanner, S. 2024. JaxPlan and GurobiPlan: Optimization baselines for replanning in discrete and mixed discrete-continuous probabilistic domains. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, 230–238.

Grathwohl, W.; Choi, D.; Wu, Y.; Roeder, G.; and Duvenaud, D. 2018. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. In *ICLR*.

Greensmith, E.; Bartlett, P. L.; and Baxter, J. 2004. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov): 1471–1530.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, 1861–1870. Pmlr.

Hafner, D.; Lillicrap, T.; Ba, J.; and Norouzi, M. 2020. Dream to control: Learning behaviors by latent imagination. In *ICLR*.

Hájek, P. 2001. *Metamathematics of fuzzy logic*, volume 4. Springer Science & Business Media.

Heiden, E.; Millard, D.; Coumans, E.; Sheng, Y.; and Sukhatme, G. S. 2021. NeuralSim: Augmenting differentiable simulators with neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 9474–9481. IEEE.

Hinton, G. 2012. Neural Networks for Machine Learning, Lecture 6e: RMSProp. Coursera lecture.

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

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*.

Nocedal, J.; and Wright, S. J. 2006. *Numerical optimization*. Springer.

Padhy, N. P. 2004. Unit commitment-a bibliographical survey. *IEEE Transactions on power systems*, 19(2): 1196–1205.

Pascanu, R.; Mikolov, T.; and Bengio, Y. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, 1310–1318. Pmlr.

Pathak, D.; Agrawal, P.; Efros, A. A.; and Darrell, T. 2017. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, 2778–2787. PMLR.

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

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

Song, Y.; Kim, S. b.; and Scaramuzza, D. 2025. Learning Quadruped Locomotion Using Differentiable Simulation. In Agrawal, P.; Kroemer, O.; and Burgard, W., eds., *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, 258–271. PMLR.

Sutton, R. S.; Barto, A. G.; et al. 1998. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge.

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

Taitler, A.; Alford, R.; Espasa, J.; Behnke, G.; Fišer, D.; Gimelfarb, M.; Pommerening, F.; Sanner, S.; Scala, E.; Schreiber, D.; Segovia-Aguas, J.; and Seipp, J. 2024. The 2023 International Planning Competition. *AI Magazine*, 45(2): 280–296.

Taitler, A.; Gimelfarb, M.; Jeong, J.; Gopalakrishnan, S.; Mladenov, M.; Liu, X.; and Sanner, S. 2022. pyrddl-gym: From rddl to gym environments. *arXiv preprint arXiv:2211.05939*.

Tiğrek, T.; Dasgupta, S.; and Smith, T. F. 2002. Nonlinear optimal control of HVAC systems. *IFAC Proceedings Volumes*, 35(1): 149–154.

Tucker, G.; Mnih, A.; Maddison, C. J.; Lawson, J.; and Sohl-Dickstein, J. 2017. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*, 30.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3): 229–256.

Wu, G.; Say, B.; and Sanner, S. 2017. Scalable planning with tensorflow for hybrid nonlinear domains. *Advances in Neural Information Processing Systems*, 30.

Yeh, W. W.-G. 1985. Reservoir management and operations models: A state-of-the-art review. *Water resources research*, 21(12): 1797–1818.

Yin, M.; and Zhou, M. 2019. ARM: Augment-REINFORCE-merge gradient for stochastic binary networks. In *ICLR*.