Autoregressive Diffusion World Models for Off-Policy Evaluation of LLM Agents

Kaixuan Liu
Emory University
kaixuan.liu@emory.edu

Guojun Xiong*
Shanghai Jiao Tong University
gjxiong@sjtu.edu.cn

Weinan Zhang
Shanghai Jiao Tong University
wnzhang@sjtu.edu.cn

Shengpu Tang
Emory University
shengpu.tang@emory.edu

Abstract

Evaluating large language model (LLM) agents in multi-turn interactive environments is expensive and risky, as it requires online environment interaction. We propose ADWM (Autoregressive Diffusion World Model), an evaluation framework that estimates the performance of a new LLM agent policy purely from pre-collected trajectories. The core idea is to learn a latent diffusion world model that simulates how the environment responds to the evaluation policy, without ever executing it in the real environment. Existing diffusion-based OPE methods guide full trajectories in a single pass by jointly diffusing states and actions, an assumption that breaks down for LLM agents whose actions are discrete text that must be sampled from the policy after observing the environment. Unlike textual world models that generate each observation token by token, ADWM jointly denoises the complete next observation in latent space, enabling step-by-step rollouts where the world model and agent alternate in causal order. Crucially, the LLM agent under evaluation directly guides the diffusion generation at each step via a policy-conditioned score function, ensuring that simulated trajectories accurately reflect its decision-making patterns. Empirically, ADWM achieves accurate policy rankings and evaluation reliability across diverse multi-turn agent tasks, demonstrating its promise as a practical framework for offline LLM agent evaluation. Code is available at https://github.com/kaixuanliu-ai/ADWM.

1 Introduction

Large language model agents are increasingly deployed in multi-turn interactive environments, navigating websites, writing and executing code, and reasoning over long document contexts [41, 44, 21]. As these agents are trusted with higher-stakes tasks, evaluating a new agent policy before deployment becomes critical. Yet evaluation is expensive; every new agent must be executed live in the real environment, consuming API budget and potentially causing irreversible side effects.

Off-policy evaluation (OPE) addresses this challenge by estimating the value of a new agent from pre-collected offline data, without any further environment interaction. Importance sampling methods reweight offline trajectories by policy likelihood ratios [25, 28], but these weights grow exponentially with trajectory length, making the estimator impractical for multi-turn agentic settings. Value-based direct methods attempt to fit a value function on offline data [24, 31], but suffer from compounding bias when the evaluation policy differs substantially from the behavior policy. Doubly robust methods

*Correspondence to Guojun Xiong <gjxiong@sjtu.edu.cn>.

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

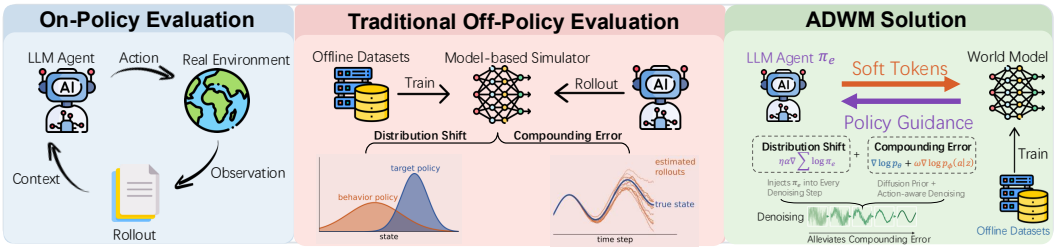


Figure 1: **Comparison of evaluation paradigms for LLM agents.** *(Left)* On-policy evaluation requires executing the agent in the real environment, which is expensive and potentially unsafe. *(Middle)* Traditional off-policy evaluation learns a model-based simulator from offline data, but suffers from two fundamental issues: distribution shift between the behavior and target policies, and compounding error accumulated over multi-step rollouts. *(Right)* Our ADWM addresses both issues simultaneously. Policy guidance injects π_e into every denoising step to alleviate distribution shift, while the diffusion prior combined with action-aware denoising alleviates compounding error.

combine the two [7, 23], partially mitigating these issues but not resolving the fundamental difficulty of long-horizon distribution shift in high-dimensional text spaces.

Among direct methods, a more promising model-based alternative is to build a world model: learn the environment dynamics from offline data and simulate rollouts under the evaluation policy [11, 26]. In the agentic setting, this simulation must be *autoregressive*: each LLM action depends on the observation just received, so the world model and agent must alternate step by step in causal order. Autoregressive transformer-based world models [29] follow this structure naturally, but generate observations token by token, causing errors to accumulate within each step and compound further across steps, a critical failure mode for the long-horizon tasks that LLM agents face. Diffusion models offer a principled remedy: by jointly denoising the complete next observation, they remove the token-level error chain within each step [14, 5], and score-function guidance enables the evaluation policy to steer rollouts without retraining. However, existing diffusion-based methods for offline RL and OPE [17, 26, 9] were designed for continuous control, where states and actions are both real-valued vectors that can be jointly noised and denoised as a single tensor in one pass, with actions generated by the diffusion process itself and the evaluation policy entering only as a guidance signal. In LLM agent settings, this assumption fundamentally breaks down: actions are discrete text that must be sampled from the LLM *after* observing the environment and cannot participate in a joint continuous diffusion over trajectories. Generating full trajectories in a single pass therefore requires all LLM actions to be known before any observation is generated, a circular dependency that is irreconcilable with the step-by-step rollout that agentic evaluation demands.

We propose **ADWM** (Autoregressive Diffusion World Model), an offline evaluation framework for LLM agents that resolves this tension by instantiating the world model itself as a diffusion process (see Figure 1). The key insight is that the globally policy-guided trajectory law can be *exactly* factored into a sequence of single-step conditionals, each decomposing as a product of a prior, an action-posterior, and a policy continuation factor (Theorem 1). This product structure maps directly onto guided diffusion, with the evaluation policy steering every denoising step via its log-likelihood gradient. Because each next observation is denoised jointly, ADWM avoids the within-step token-level error chain, although transition errors may still propagate across steps. ADWM is trained entirely offline, and a new agent only requires fitting a lightweight adapter.

Our contributions are summarized as follows:
▷ We derive the exact autoregressive single-step conditional induced by a policy-guided trajectory law, showing it decomposes into a principled three-factor product (Theorem 1).
▷ We realize this decomposition as a guided diffusion process in a structured latent space, where the evaluation policy participates in every denoising step without retraining the world model.
▷ We demonstrate empirically that ADWM correctly ranks evaluation policies across diverse LLM agent benchmarks, outperforming classic OPE baselines under realistic LLM importance ratios.

2 Related Work

Model-Based OPE. Model-based OPE methods estimate policy value by learning a transition model from offline data and simulating rollouts under the evaluation policy [42, 43]. The core difficulty is compounding model error: small inaccuracies in the learned transition accumulate across steps, causing simulated trajectories to drift away from the true environment dynamics over long horizons [18, 35, 2]. Existing approaches mitigate this through pessimistic value penalties or conservative policy optimization [22], but these techniques were designed for low-dimensional continuous control and do not extend to free-form text observations. Beyond error accumulation, these methods provide no mechanism for conditioning the simulated rollout on the evaluation policy, so generated trajectories reflect the behavior distribution rather than what the evaluation policy would actually produce [36, 8]. ADWM addresses both issues: jointly denoising each next observation removes the token-level error chain within a step [19, 1], while the guided score function ensures every generated observation is conditioned on the evaluation policy.

Diffusion Models for World Modeling. Diffusion models have emerged as powerful environment simulators that generate each transition through an iterative denoising process rather than token by token. Full-sequence models jointly generate entire trajectories [19, 1, 15], treating states and actions as a single jointly diffused tensor. This design precludes step-by-step rollout, as actions are produced by the diffusion process itself rather than sampled from an external policy. Score-based guidance [13] can steer such generation toward a target policy [26, 17], but inherits the same limitation: actions must be real-valued vectors that participate in the joint diffusion, an assumption that breaks down for LLM agents whose actions are discrete text sampled from the policy after observing the environment. ADWM resolves this by deriving an exact factored score that enables the evaluation policy to steer each denoising step autoregressively, without requiring actions to be part of the diffusion process. Autoregressive video diffusion models such as Diffusion Forcing [3] and Self Forcing [16] generate frames sequentially by assigning per-frame noise levels and by training on self-generated histories, respectively. These methods study sequential generation within a single diffusion model. ADWM instead alternates a diffusion world model with an external evaluation policy whose action is available only after it observes the current observation, so the interaction trajectory cannot be jointly diffused without violating causal order.

Textual World Models for LLM Agents. LLMs themselves can act as world models that predict environment feedback in text. SURGE [27] studies LLMs as surrogate code executors, CWM [6] trains a code LLM on execution traces and agentic interactions, and RAP [12] uses an LLM world model for planning during reasoning. These simulators generate each observation token by token, so an early token error changes the prefix of all remaining tokens within the same environment step. Moreover, they target code execution, environment simulation, or online planning rather than OPE. ADWM learns from offline behavior trajectories, denoises the complete next-observation latent jointly, and estimates the value of a distinct evaluation policy without additional real-environment interaction.

3 Problem Formulation

3.1 Agentic Reinforcement Learning

We formulate agentic reinforcement learning as a sequential decision-making problem, modeled as a Partially Observable Markov Decision Process (POMDP) $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P, \mathcal{E}, r, \mu, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{O}$ is the observation space, $P : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition kernel, $\mathcal{E} : \mathcal{S} \to \Delta(\mathcal{O})$ is the emission distribution, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\mu \in \Delta(\mathcal{S})$ is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor. Since the agent does not have direct access to s_t, it instead receives a partial observation $o_t \sim \mathcal{E}(\cdot \mid s_t)$ emitted by the environment. The agent interacts with the environment over T turns. At each turn $t \in \{1, \ldots, T\}$, the environment is in state $s_t \in \mathcal{S}$, from which the agent receives observation $o_t \in \mathcal{O}$ and selects an action $a_t \sim \pi(\cdot \mid h_t)$ based on the interaction history:

$$h_t = (o_1, a_1, \ldots, o_{t-1}, a_{t-1}, o_t).$$

The environment then transitions to the next state $s_{t+1} \sim P(\cdot \mid s_t, a_t)$ and emits a reward $r_t = r(s_t, a_t)$. A complete trajectory $\tau = (o_1, a_1, r_1, \ldots, o_T, a_T, r_T)$ has discounted return

$R(\tau) = \sum_{t=1}^{T} \gamma^{t-1} r_t$, and induces the trajectory distribution:

$$p_\pi(\tau) = \int \mu(s_1) \prod_{t=1}^{T} \mathcal{E}(o_t \mid s_t)\, \pi(a_t \mid h_t)\, P(s_{t+1} \mid s_t, a_t)\, ds_{1:T+1}.$$

The value of policy π is thus $J(\pi) = \mathbb{E}_{\tau \sim p_\pi}[R(\tau)]$. For example, in WebShop [40], o_t is the current web page together with the task instruction, the LLM reads h_t and outputs a text action a_t such as `search[...]` or `click[buy now]`, and the environment returns the next page o_{t+1}, the reward r_t, and a termination signal. A logged episode annotated with this notation is given in Appendix F.11.

3.2 Off-Policy Evaluation for LLM Agents

We assume access to an offline dataset:

$$\mathcal{D} = \{\tau^{(i)}\}_{i=1}^{N}, \qquad \tau^{(i)} \sim p_{\pi_b}(\tau),$$

collected by a behavior policy π_b, which may correspond to a mixture of different LLM configurations and is treated as unknown. Given $\mathcal{D}$ and a target evaluation policy π_e, a language model distinct from π_b that maps observation history h_t to a distribution over text actions, the goal of off-policy evaluation (OPE) is to estimate:

$$J(\pi_e) = \mathbb{E}_{\tau \sim p_{\pi_e}}[R(\tau)],$$

without executing π_e in the real environment. The core challenge is distribution shift: since $p_{\pi_b}(\tau) \neq p_{\pi_e}(\tau)$, trajectories in $\mathcal{D}$ are not representative samples from p_{π_e}, and direct Monte Carlo estimation on $\mathcal{D}$ is severely biased. We address this challenge by learning a diffusion world model that simulates rollouts under π_e directly, bypassing the need for explicit density ratio estimation.

3.3 Denoising Diffusion Models

Denoising Diffusion Probabilistic Models (DDPMs) [14] learn to approximate a target distribution $q(x)$ via two coupled Markov chains. The forward process gradually corrupts a clean sample $x^0 \sim q(x)$ over K steps:

$$x^k = \sqrt{\bar{\alpha}_k}\, x^0 + \sqrt{1 - \bar{\alpha}_k}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I),$$

where $\{\bar{\alpha}_k\}_{k=1}^{K}$ is a decreasing noise schedule with $\bar{\alpha}_K \approx 0$. The reverse process learns to recover x^0 from x^K by iteratively applying $p_\theta(x^{k-1} \mid x^k) = \mathcal{N}(x^{k-1};\, \mu_\theta(x^k, k),\, \sigma_k^2 I)$, where μ_θ is parameterized via a noise prediction network ϵ_θ trained by:

$$\mathcal{L}(\theta) = \mathbb{E}_{k,\, x^0,\, \epsilon \sim \mathcal{N}(0,I)} \left[\|\epsilon - \epsilon_\theta(x^k, k)\|^2 \right].$$

Guided diffusion. A trained diffusion model can be steered to sample from a conditional distribution $p(x \mid y)$ without retraining, by incorporating a guidance signal y into each denoising step [5]. Since the backward transition is well approximated by a Gaussian, $p(x^k \mid x^{k+1}) \approx \mathcal{N}(\mu_k, \Sigma_k)$, Bayes' rule gives:

$$p(x^k \mid x^{k+1}, y) \propto p(x^k \mid x^{k+1})\, p(y \mid x^k).$$

Applying a first-order Taylor expansion of $\log p(y \mid x^k)$ around μ_k yields:

$$p(x^k \mid x^{k+1}, y) \approx \mathcal{N}\big(x^k;\ \mu_k + \Sigma_k g_k,\ \Sigma_k\big), \qquad g_k = \nabla_x \log p(y \mid x)\big|_{x=\mu_k}.$$

Sampling from the guided distribution therefore reduces to shifting the denoising mean by $\Sigma_k g_k$ at each step, with no modification to the model itself. In our framework, y is instantiated as the target LLM agent π_e, and g_k becomes the log-likelihood gradient of π_e with respect to the current latent state, steering world-model rollouts toward trajectories compatible with the agent under evaluation.

4 Autoregressive Diffusion World Models

We develop an autoregressive diffusion world model for off-policy evaluation of LLM agents. The core insight is that a globally policy-guided trajectory law can be exactly factored into a sequence of single-step conditionals, each fusing local action consistency with global policy guidance. We realize these conditionals as a guided diffusion process in a structured latent space, where the evaluation LLM agent directly steers the denoising at every step (Figure 2).

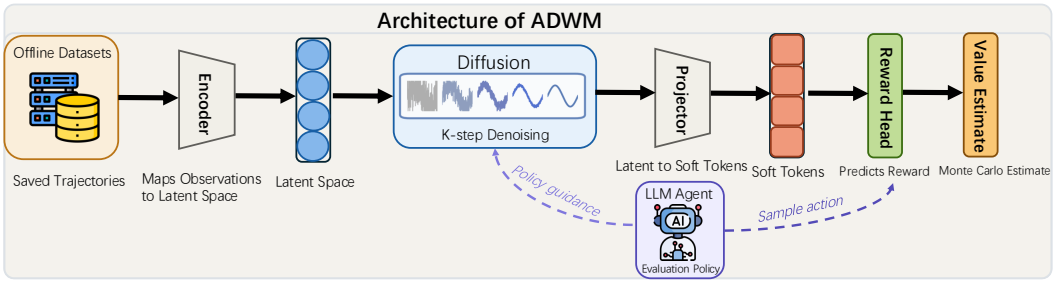


Figure 2: **Architecture of ADWM.** Offline trajectories are encoded by E into latent states z_t, which are processed by a diffusion world model p_θ through K-step denoising. A projector G_ψ maps each latent to soft tokens $\tilde{o}_t$ that the evaluation policy π_e can read in its own embedding space. π_e plays two complementary roles: it steers the denoising process via policy guidance (dashed arrow, Section 4.2), and samples actions a_t conditioned on $\tilde{o}_t$ to drive autoregressive rollout (solid arrow). A reward head r_ρ predicts rewards $\hat{r}_t$ used in the Monte Carlo value estimate $\hat{J}(\pi_e)$.

4.1 From Trajectory Guidance to Single-Step Conditionals

In our agentic setting, the behavior policy π_b is unknown and generally intractable, as trajectories in $\mathcal{D}$ may originate from different LLM configurations, prompting strategies, or agent versions. A natural starting point is importance sampling, which corrects for the mismatch between p_{π_b} and p_{π_e} via policy-likelihood ratios. Since the environment transition cancels between p_{π_e} and p_{π_b}, the target trajectory law satisfies:

$$p_{\pi_e}(\tau) = p_{\pi_b}(\tau) \cdot \prod_{t=1}^{T} \frac{\pi_e(a_t \mid h_t)}{\pi_b(a_t \mid h_t)}, \tag{1}$$

so $\log p_{\pi_e}(\tau) = \log p_{\pi_b}(\tau) + \sum_t \log \pi_e(a_t \mid h_t) - \sum_t \log \pi_b(a_t \mid h_t)$. Since π_b is unavailable, and dropping it acts as a form of behavior regularization [17] that anchors the distribution to the support of $\mathcal{D}$, we learn a generative model $p_\theta(\tau)$ of the logged trajectory distribution via maximum likelihood, and introduce $\alpha > 0$ to control the correction strength, yielding the policy-guided target law:

$$q_\alpha(\tau \mid \pi_e) \;\propto\; p_\theta(\tau)\, \exp\Big(\alpha \sum_{t=1}^{T} \log \pi_e(a_t \mid h_t)\Big), \qquad \alpha > 0. \tag{2}$$

When $p_\theta = p_{\pi_b}$ and $\alpha = 1$, this recovers p_{π_e} exactly; the full derivation is in Appendix B.

The target law (2) is defined over full trajectories, but cannot be instantiated directly for a black-box π_e. Since π_e is a black-box LLM that must observe o_t before producing a_t, the trajectory-level tilt $\sum_t \log \pi_e(a_t \mid h_t)$ cannot be evaluated without first generating all observations in causal order. This forces an autoregressive decomposition: the world model and π_e must alternate step by step, generating observations and actions one at a time. The following proposition, proved in Appendix C, characterizes the exact single-step conditional induced by (2).

Proposition 1 (Autoregressive conditional law)**.** *Let q_α be the policy-guided trajectory law (2). Then for every (h_t, a_t):*

$$q_\alpha(o_{t+1} \mid h_t, a_t; \pi_e) \;\propto\; p_\theta(o_{t+1} \mid h_t, a_t)\, \mathcal{C}_\alpha(o_{t+1}; h_t, a_t, \pi_e), \tag{3}$$

where the continuation factor *is:*

$$\mathcal{C}_\alpha(o_{t+1}; h_t, a_t, \pi_e) \;:=\; \mathbb{E}_{\tau_{t+1:T} \sim p_\theta(\cdot \mid h_t, a_t, o_{t+1})} \left[\exp\Big(\alpha \sum_{u=t+1}^{T} \log \pi_e(a_u \mid h_u)\Big) \right]. \tag{4}$$

Two candidates for o_{t+1} may be equally supported by $p_\theta(o_{t+1} \mid h_t, a_t)$ yet lead to futures with very different compatibility with π_e; $\mathcal{C}_\alpha$ is precisely the term that distinguishes them. A Bayesian decomposition of $p_\theta(o_{t+1} \mid h_t, a_t)$ further separates the prior dynamics from the influence of a_t, giving the central result of our framework.

Theorem 1 (Fused single-step conditional)**.** *Let q_α be the policy-guided trajectory law (2). For every (h_t, a_t) and any $\omega > 0$, $\eta \geq 0$:*

$$\widetilde{q}_{\omega,\eta}(o_{t+1} \mid h_t, a_t; \pi_e) \propto \underbrace{p_\theta(o_{t+1} \mid h_t)}_{\text{prior}} \cdot \underbrace{p_\theta(a_t \mid o_{t+1}, h_t)^\omega}_{\text{action posterior}} \cdot \underbrace{\mathcal{C}_\alpha(o_{t+1}; h_t, a_t, \pi_e)^\eta}_{\text{policy continuation}}, \tag{5}$$

where $\omega = \eta = 1$ recovers the exact conditional induced by q_α.

Remark 1 (Limiting cases of ω and η)**.** *The parameters ω and η interpolate between degenerate regimes that illuminate the role of each guidance term. Setting $\eta = 0$ removes the policy continuation factor entirely, reducing $\widetilde{q}_{\omega,0}$ to an action-posterior-weighted prior that simulates the behavior distribution with no correction toward π_e. Setting $\omega = 0$ suppresses the action posterior, causing the model to generate observations from the unconditional prior $p_\theta(o_{t+1} \mid h_t)$ irrespective of a_t and breaking action-consistency. The canonical choice $\omega = \eta = 1$ recovers the exact conditional induced by q_α. In practice, ω is controlled by the CFG scale λ via $\omega = 1 + \lambda$, and η is annealed over the reverse process; both are treated as hyperparameters rather than tuned per environment.*

The three factors play distinct roles. The prior $p_\theta(o_{t+1} \mid h_t)$ anchors generated observations to the support of the offline data. The action posterior $p_\theta(a_t \mid o_{t+1}, h_t)^\omega$ injects current-step action information without a separate classifier, recovered directly from classifier-free guidance as shown in Section 4.2. The continuation factor $\mathcal{C}_\alpha$ re-weights candidate observations by their long-horizon compatibility with π_e, ensuring the world model generates observations that open up futures where π_e can act effectively. Crucially, since (5) is a single-step conditional, π_e never needs to produce actions before observations are generated, fully resolving the circular dependency that prevents existing diffusion OPE methods from operating autoregressively. The proof is given in Appendix D. Each factor corresponds to a distinct gradient term in the score function, which we now realize as a guided diffusion process in latent space.

Since (5) is a product of three factors, its log decomposes into a sum, and differentiating with respect to o_{t+1} yields an additive score:

$$\nabla_{o_{t+1}} \log \widetilde{q}_{\omega,\eta} = \underbrace{\nabla \log p_\theta(o_{t+1} \mid h_t)}_{\text{prior score}} + \omega \underbrace{\nabla \log p_\theta(a_t \mid o_{t+1}, h_t)}_{\text{action-posterior score}} + \eta \underbrace{\nabla \log \mathcal{C}_\alpha(o_{t+1}; h_t, a_t, \pi_e)}_{\text{continuation score}}. \tag{6}$$

This additive structure is precisely what makes guided diffusion applicable: each term can be approximated and injected independently into the reverse process without retraining the world model. The prior score is given directly by the score network; the action-posterior score is recovered from the difference between action-conditioned and unconditional outputs via classifier-free guidance; and the continuation score requires differentiating through π_e's log-likelihoods along future rollouts. We realize all three in a structured latent space, as detailed next.

4.2 Guided Diffusion World Model

The product structure of Theorem 1 maps directly onto guided diffusion: each factor contributes an additive term to the score, and sampling reduces to running a guided reverse process with π_e steering the denoising at every step. To instantiate this in practice, we first specify the latent space in which the diffusion operates and how π_e reads world-model states.

Latent state and policy interface. Observations in LLM agent tasks are raw text, which cannot be directly denoised. We therefore operate in a structured latent space. An encoder E maps each observation to a compact representation $z_t = E(o_t) \in \mathbb{R}^d$, and a projector G_ψ produces soft tokens $\tilde{o}_t = G_\psi(z_t) \in \mathbb{R}^{n_{\text{soft}} \times d_{\text{LLM}}}$ that π_e can read in its own embedding space. The world model operates on the latent history $h_t = (z_1, a_1, \ldots, z_t)$, while π_e conditions on $h_t^\pi = (\tilde{o}_1, a_1, \ldots, \tilde{o}_t)$. A pretrained semantic encoder is insufficient for this role: two observations that differ only in a negation may be nearby in semantic space yet lead to entirely different dynamics, corrupting all three guidance terms in (9) simultaneously. We therefore train E end-to-end jointly with the world-model components on $\mathcal{D}$, so that the representation is shaped by the dynamics of the environment rather than surface semantics. The projector G_ψ is then fitted to the evaluation LLM with a contrastive InfoNCE objective [30]:

$$\mathcal{L}_\psi = -\frac{1}{2B} \sum_{i=1}^{B} \left[\log \frac{e^{s_{ii}/\kappa}}{\sum_j e^{s_{ij}/\kappa}} + \log \frac{e^{s_{ii}/\kappa}}{\sum_j e^{s_{ji}/\kappa}} \right] + \frac{\beta}{B} \sum_{i=1}^{B} \left\| \bar{G}_\psi(z_i) - \bar{e}_i \right\|_2^2, \tag{7}$$

where $s_{ij} = \mathrm{cossim}(\bar{G}_\psi(z_i), \bar{e}_j)$, $\bar{G}_\psi(z_i)$ is the token-mean of the projected soft tokens, $\bar{e}_i$ is the token-mean of the ground-truth π_e input embeddings, κ is a temperature, and $\beta = 0.1$ balances discriminability against distributional alignment with π_e's embedding space. Two auxiliary losses shape the representation: an inverse dynamics loss $\mathcal{L}_{\mathrm{IDM}}$ encouraging action-aware latents, and a behavior cloning loss $\mathcal{L}_{\mathrm{BC}}$ providing a stable retrieval target for G_ψ; full definitions are in Appendix F.7.

Guided score. Following classifier-free guidance [13], we parameterize the world model as a score network $s_\theta(z^k_{t+1}, k; h^z_t, a_t)$. A causal Transformer encodes the latent history h^z_t from the sequence of (z_u, a_u) pairs with a subsequent-step mask, so the summary at step t depends only on past latents and actions. The score network is an MLP that receives the noised latent z^k_{t+1}, the diffusion step k, this history summary, and the action embedding through FiLM conditioning (Appendix F.6). To realize the prior and action-posterior factors of Theorem 1 without a classifier, we train s_θ under two regimes by randomly masking a_t with probability p during training:

$$\mathcal{L}_{\mathrm{DSM}}(\theta) = \mathbb{E}\Big[\big\| s_\theta(z^k_{t+1}, k; h^z_t, \tilde{a}_t) - \nabla_{z^k} \log p_k(z^k_{t+1} \mid z_{t+1}) \big\|^2\Big], \quad \tilde{a}_t = \begin{cases} \emptyset & \text{w.p. } p, \\ a_t & \text{otherwise.} \end{cases} \tag{8}$$

By Bayes' rule, the difference between the action-conditioned and unconditional outputs recovers the score of the implicit action posterior $p_\phi(a_t \mid z^k_{t+1}, h^z_t)$, so classifier-free guidance with scale λ directly realizes the first two terms of (5), with $\omega = 1 + \lambda$.

For the continuation term, the score $\nabla_{z_{t+1}} \log \mathcal{C}_\alpha$ requires an expectation over all future trajectories and is intractable. At each diffusion step k, the score network provides a denoised estimate $\hat{z}_{t+1}$; we roll the world model forward from this estimate and differentiate through π_e's log-likelihoods along the resulting trajectory. The complete guided score is:

$$\begin{aligned} s(z^k_{t+1}, k; h^z_t, a_t, \pi_e) = {} & \nabla_{z^k_{t+1}} \log p_\theta(z^k_{t+1} \mid h^z_t) + \omega \, \nabla_{z^k_{t+1}} \log p_\phi(a_t \mid z^k_{t+1}, h^z_t) \\ & + \eta_k \, \alpha \, \nabla_{\hat{z}_{t+1}} \sum_{u=t+1}^{T} \log \pi_e(\hat{a}_u \mid h^\pi_u), \end{aligned} \tag{9}$$

where η_k is annealed so that continuation guidance strengthens as $\hat{z}_{t+1}$ becomes more reliable late in the reverse process. The complete derivation of (9) from Theorem 1 is given in Appendix E.

Value estimation. Given the guided transition model, we estimate $J(\pi_e)$ by Monte Carlo rollout. At each step, π_e samples an action $a_t \sim \pi_e(\cdot \mid h^\pi_t)$, the world model generates the next latent z_{t+1} via the reverse diffusion process with score (9), and the reward head predicts $\hat{r}_t = r_\rho(z_{t+1}, h^z_t)$. Episodes terminate when the termination head $\hat{d}_t = \sigma(d_\rho(z_{t+1}, h^z_t))$ exceeds a threshold. Averaging over M rollouts gives:

$$\widehat{J}(\pi_e) = \frac{1}{M} \sum_{m=1}^{M} \sum_{t=1}^{T_m} \hat{r}^{(m)}_t, \tag{10}$$

where T_m is the termination step of the m-th rollout. The rollout follows the causal order of the real interaction: π_e reads only the soft tokens $\tilde{o}_{1:t}$ produced by the world model, and each next latent is generated only after a_t is sampled. Hence no environment interaction is required at evaluation time.

5 Experiments

We evaluate ADWM around three questions:

1. *Can a world model rank unseen policies from behavior data alone?*
2. *How does it compare to classical OPE under realistic LLM importance ratios?*
3. *Which components of the diffusion guidance matter?*

Experiment setup. We evaluate ADWM on four LLM-agent benchmarks chosen to span the reward and post-training regimes of modern LLM agents. On the reward axis, the suite covers dense per-step reward (HotpotQA F1 [39]), shaped partial reward (ScienceWorld [37]), continuous partial reward

(WebShop [40]), and sparse binary success (ALFWorld [33]). On the policy axis, the cells cover task-specific RLHF-style post-training (DPO and PRM on HotpotQA [38]; ETO on ScienceWorld [34]) and generic-to-specialized iterative fine-tuning (LEAP on ALFWorld and WebShop [4]). All evaluation policies are publicly released checkpoints from prior work, summarised alongside their behavior counterparts in Table 1. Representative logged interaction traces for WebShop and ALFWorld are provided in Appendices F.11 and F.12.

Table 1: Environments, policies, and reward structure. The behavior LLM π_b is used only to collect world-model training data; the evaluation LLM π_e is the policy ADWM scores at test time. Citations point to the benchmark source and the released π_e checkpoint. Per-cell behavior trajectory counts, GT episode budgets, and full hyperparameters are in Appendix F.

Env.	Behavior π_b	Evaluation π_e	Reward
HotpotQA [39]	ReAct-HotpotQA-SFT	ReAct-HotpotQA {DPO, PRM} [38]	dense F1
ScienceWorld [37]	sw-llama-sft	sw-llama-eto [34]	partial
ALFWorld [33]	Llama-3.1-8B-Instr.	leap-alf-{iter1, iter3} [4]	sparse 0/1
WebShop [40]	Llama-3.1-8B-Instr.	leap-ws-iter1 [4]	cont. partial

In every cell, the behavior policy π_b used to collect world-model training trajectories is strictly distinct from the evaluation policy π_e being scored. Concretely, π_b is either the matching task-specific SFT base of the same agent (HotpotQA, ScienceWorld) or, when no such base has been publicly released, the generic instruction-tuned Llama-3.1-8B-Instruct [10] (ALFWorld, WebShop); the ALFWorld-iter3 and WebShop pools also include two other LEAP iterates (Table 3). The world model is therefore trained without ever observing π_e. On ALFWorld we additionally evaluate the third LEAP iteration to test cross-iteration ranking; on HotpotQA we pool DPO and PRM checkpoints into a 10-cell cross-policy split that probes policy-level discrimination beyond the within-policy ε-axis.

To assess sensitivity across a graded range of policy quality, we construct a one-parameter family of evaluation policies via ε-greedy mixing: at each step, π_e acts with probability $1-\varepsilon$ and a uniformly random admissible action is taken with probability ε, for $\varepsilon \in \{0, 0.25, 0.5, 0.75, 1.0\}$. The real-environment ground-truth curve is obtained by executing π_e^{ε} in the actual benchmark; ADWM produces $\hat{J}$ entirely from world-model rollouts with no environment access at evaluation time. We run five random seeds per cell and report the Spearman rank correlation between the seed-averaged $\hat{J}$ curve and the ground-truth curve. Full experimental details are in Appendix F.

The behavior and evaluation policies are not close. We measure the shift by the cumulative trajectory log-likelihood ratio:

$$\log \frac{p_{\pi_e}(\tau)}{p_{\pi_b}(\tau)} = \sum_{t=1}^{T} \log \frac{\pi_e(a_t \mid h_t)}{\pi_b(a_t \mid h_t)}.$$

At ε=0.5, its magnitude reaches up to 24, 18, 35, and 28 nats on HotpotQA, ScienceWorld, ALFWorld, and WebShop, so the trajectory likelihood ratios span roughly 16 to 30 orders of magnitude (Appendix G.3). The evaluation checkpoint is also excluded from every behavior pool. The experiments therefore go beyond the near-policy regime, although they do not cover fully out-of-support or cross-family evaluation.

Baselines. We compare ADWM against five classical OPE estimators: direct method (DM) [36], importance sampling (IS) [32], weighted importance sampling (WIS) [32], fitted-Q evaluation (FQE) [24], and doubly robust estimation (DR) [20], all implemented following the COBS reference protocol [36]. For estimators that require π_b (IS, WIS, DR), we supply token log-probabilities of the logged actions under the behavior LLM rather than the uniform action prior commonly used in prior work, providing these baselines with privileged information that is unavailable in realistic deployments (closed-source APIs, deleted checkpoints, hidden tokenizers). We include them as diagnostic references rather than competitive baselines, since the unbounded LLM action space and limited distributional overlap in long-horizon rollouts violate the absolute-continuity assumption underlying IS-based estimators. Implementation details and the per-ε FQE correction are in Appendix G.

5.1 Results

Main results. Table 2 compares ADWM against all five classical estimators across six (π_b, π_e) configurations. ADWM is the only estimator with strictly positive Spearman ρ in every cell, ranging from $+0.67$ on ALFWorld-iter1 to $+0.90$ on both HotpotQA-DPO and WebShop-iter1 (mean $+0.82$), while every classical baseline is non-positive in at least two cells. IS and DR collapse under LLM-induced importance ratios spanning up to thirty orders of magnitude, with correlations as low as -0.90. WIS eliminates the explosion but degenerates to a single dominant-weight trajectory over long horizons, producing $\rho{=}0$ on both ALFWorld cells and spurious positive correlations elsewhere driven by floating-point underflow ties. DM estimates a behavior-policy state value invariant to the evaluation policy's action distribution, yielding $\rho{=}0$ within every ε-sweep and inverting on the cross-policy split ($\rho{=}-0.45$). FQE is the sole partial exception: once corrected for the ε-linearity artifact in the COBS protocol, it attains $\rho{=}+0.82$ on ALFWorld-iter1 but collapses or inverts elsewhere (mean $\rho{=}+0.10$). ADWM, requiring neither importance weights nor a linear inductive bias, outperforms the next-best baseline by $+0.34$ in mean correlation; the world model is trained exclusively on π_b trajectories in every cell. Per-seed statistics and $\hat{J}$/GT curves are in Appendix H.

Table 2: Per-cell Spearman ρ between each estimator's $\hat{J}$ and the real-environment ground-truth curve. ADWM is the only estimator with $\rho{>}0$ on every configuration (mean $+0.82$, min $+0.67$). CI_{95} and p-values from 2000 bootstrap resamples on the $(\hat{J}, \mathrm{GT})$ pairs. Per-seed values and the underlying $\hat{J}$ / GT curves are in Appendix H.

		ADWM (ours)			Classical OPE baselines (Spearman ρ)				
Configuration	n	ρ	CI_{95}	p	FQE	DR	DM	IS	WIS
HotpotQA-DPO	5	$\mathbf{+0.90}$	$[+0.11, +1.00]$	0.037	-0.10	-0.90	0.00	-0.90	$\mathbf{+0.90}$
ScienceWorld-ETO	5	$+0.82$	$[+0.00, +1.00]$	0.089	-0.21	$+0.21$	0.00	$+0.21$	$\mathbf{+0.97}$
ALFWorld-iter1	5	$+0.67$	$[-0.91, +1.00]$	0.219	$\mathbf{+0.82}$	$+0.21$	0.00	-0.82	0.00
ALFWorld-iter3	5	$\mathbf{+0.80}$	$[+0.11, +1.00]$	0.104	0.00	$+0.40$	0.00	-0.70	0.00
WebShop-iter1	5	$\mathbf{+0.90}$	$[+0.11, +1.00]$	0.037	0.00	-0.20	0.00	-0.90	$\mathbf{+0.90}$
HotpotQA-cross	10	$\mathbf{+0.81}$	$[+0.23, +1.00]$	0.005	$+0.12$	-0.64	-0.45	-0.52	$+0.13$
Mean ρ		$\mathbf{+0.82}$			$+0.10$	-0.15	-0.07	-0.61	$+0.48$
Min ρ		$\mathbf{+0.67}$			-0.21	-0.90	-0.45	-0.90	0.00

Training convergence. Figure 3 shows that all loss components converge stably across all four benchmarks within 50 epochs. Beyond $\mathcal{L}_{\mathrm{DSM}}$ (8), the total objective $\mathcal{L}_{\mathrm{WM}}$ includes two auxiliary terms: an inverse dynamics loss $\mathcal{L}_{\mathrm{IDM}}$ that predicts the bridging action between consecutive latents, encouraging action-aware representations, and a behavior cloning loss $\mathcal{L}_{\mathrm{BC}}$ that provides a stable retrieval target for the ψ adapter. $\mathcal{L}_{\mathrm{DSM}}$ converges fastest; $\mathcal{L}_{\mathrm{IDM}}$ and $\mathcal{L}_{\mathrm{BC}}$ converge more gradually, consistent with the greater difficulty of predicting actions and cloning behavior on long-horizon text observations. Full definitions and loss weights are in Appendix F.7.

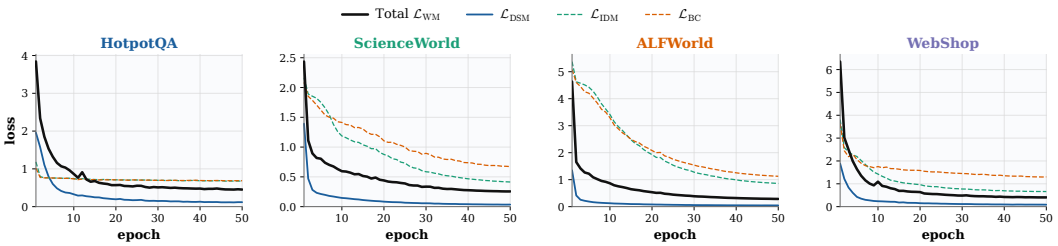


Figure 3: Training loss curves across all four benchmarks. Total world-model loss $\mathcal{L}_{\mathrm{WM}}$ and its components converge stably within 50 epochs; see Appendix F.7 for full definitions and Appendix F.8 for the training procedure.

Ablations. Figure 4 isolates the contribution of ADWM's three core components across three environments with distinct reward structures. Each component dominates in a different reward

regime, and no single component suffices across the full suite. Local CFG is most critical on long-horizon partial-reward tasks: removing it collapses WebShop ρ from $+0.90$ to $+0.10$ and drops ScienceWorld from $+1.00$ to $+0.60$, with a smaller drop on HotpotQA ($+0.90 \rightarrow +0.70$). Continuation guidance is the primary mechanism for keeping trajectories on a successful path when per-step reward is absent, with removal degrading WebShop to $+0.30$ and ScienceWorld to $+0.90$. The ψ adapter dominates on linguistically rich environments: its removal drops HotpotQA to $+0.40$ and degrades both ScienceWorld and WebShop to $+0.60$. ADWM's robustness therefore requires the combination of all three components. Additional ablations on the latent dimension and on the design of the ψ adapter loss are given in Appendix I.

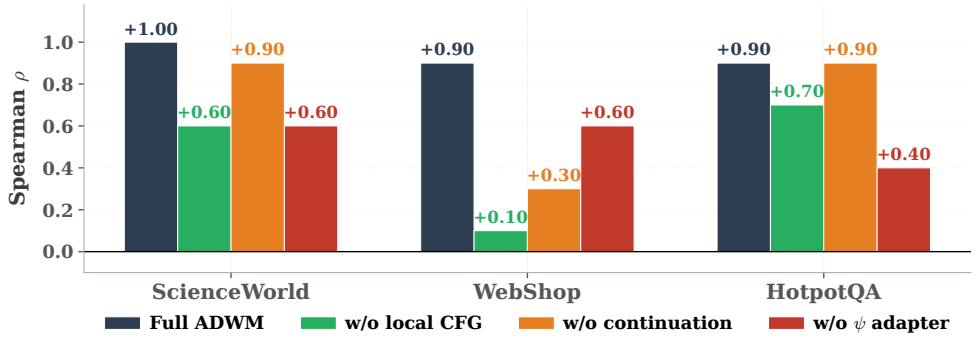


Figure 4: Component ablation on three environments (avg-$\hat{J}$ Spearman ρ, single seed; five-seed trends in Appendix I). Each component dominates in a distinct reward regime: local CFG on long-horizon partial-reward tasks, continuation guidance on goal-conditioned trajectories, and the ψ adapter on linguistically rich state spaces. Removing any single component degrades at least one environment; full ADWM consistently achieves the highest ρ.

6 Limitations

The policy-ranking experiments rely mainly on ε-greedy variants of a small number of publicly released base policies, because released agent checkpoints with different performance levels on the same benchmark remain scarce. Spearman ρ measures ranking rather than the calibration of absolute values, and we do not ablate the size or the behavior-policy diversity of the offline data. ADWM removes the token-level error chain within each observation, but transition errors may still propagate across environment steps, and we do not provide an end-to-end error bound. Finally, the soft-token projector is fitted to each evaluation LLM, although neither the offline trajectories nor the transition model need to be recollected or retrained. A natural extension is a hybrid protocol that uses ADWM for offline ranking and spends a small online budget only on the highest-ranked or hardest-to-distinguish policies. Details are in Appendix A.

7 Conclusion

We presented ADWM, an offline evaluation framework that estimates LLM agent policy value purely from pre-collected trajectories. The key contribution is an exact decomposition of a policy-guided trajectory law into autoregressive single-step conditionals, each realized as a guided diffusion process in which the evaluation policy steers every denoising step without retraining or explicit importance weights. Empirically, ADWM is the only estimator with consistently positive rank correlations across all six (π_b, π_e) configurations, with ablations confirming that robustness requires the combination of all three guidance components.

Acknowledgments and Disclosure of Funding

This work was supported in part by the National Natural Science Foundation of China (62322603) and the Shanghai Municipal Science and Technology Major Project (2025SHZDZX025D08) (to WZ), and the Emory URC Research Award and the Disruptive Discovery Seed Program by Emory

University's Office of the Provost (to ST). The content in this document are those of the authors and should not be interpreted as necessarily representing the views of the funding sources. The authors declare no competing interests.

References

[1] Anurag Ajay, Yilun Du, Abhi Gupta, Joshua Tenenbaum, Tommi Jaakkola, and Pulkit Agrawal. Is conditional generative modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.

[2] Kavosh Asadi, Dipendra Misra, Seungchan Kim, and Michel L Littman. Combating the compounding-error problem with a multi-step model. *arXiv preprint arXiv:1905.13320*, 2019.

[3] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[4] Sanjiban Choudhury and Paloma Sodhi. Better than your teacher: Llm agents that learn from privileged ai feedback. *arXiv preprint arXiv:2410.05434*, 2024.

[5] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[6] FAIR CodeGen Team. CWM: An open-weights LLM for research on code generation with world models. *arXiv preprint arXiv:2510.02387*, 2025.

[7] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. More robust doubly robust off-policy evaluation. In *International Conference on Machine Learning*, pages 1447–1456. PMLR, 2018.

[8] Yihao Feng, Tongzheng Ren, Ziyang Tang, and Qiang Liu. Accountable off-policy evaluation with kernel bellman statistics. In *International Conference on Machine Learning*, pages 3102–3111. PMLR, 2020.

[9] Hossein Goli, Michael Gimelfarb, Nathan Samuel de Lara, Haruki Nishimura, Masha Itkina, and Florian Shkurti. Stitch-ope: Trajectory stitching with guided diffusion for off-policy evaluation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[10] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[11] Josiah Hanna, Peter Stone, and Scott Niekum. Bootstrapping with models: Confidence intervals for off-policy evaluation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[12] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, 2023.

[13] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[14] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[15] Siqiao Huang, Jialong Wu, Qixing Zhou, Shangchen Miao, and Mingsheng Long. Vid2world: Crafting video diffusion models to interactive world models. *arXiv preprint arXiv:2505.14357*, 2025.

[16] Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

[17] Matthew Thomas Jackson, Michael Tryfan Matthews, Cong Lu, Benjamin Ellis, Shimon Whiteson, and Jakob Foerster. Policy-guided diffusion. *arXiv preprint arXiv:2404.06356*, 2024.

[18] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32, 2019.

[19] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[20] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International conference on machine learning*, pages 652–661. PMLR, 2016.

[21] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? *arXiv preprint arXiv:2310.06770*, 2023.

[22] Ying Jin, Zhuoran Yang, and Zhaoran Wang. Is pessimism provably efficient for offline rl? In *International conference on machine learning*, pages 5084–5096. PMLR, 2021.

[23] Nathan Kallus and Masatoshi Uehara. Efficiently breaking the curse of horizon in off-policy evaluation with double reinforcement learning. *Operations Research*, 70(6):3282–3302, 2022.

[24] Hoang Le, Cameron Voloshin, and Yisong Yue. Batch policy learning under constraints. In *International Conference on Machine Learning*, pages 3703–3712. PMLR, 2019.

[25] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. *Advances in neural information processing systems*, 31, 2018.

[26] Cong Lu, Philip Ball, Yee Whye Teh, and Jack Parker-Holder. Synthetic experience replay. *Advances in Neural Information Processing Systems*, 36:46323–46344, 2023.

[27] Bohan Lyu, Siqiao Huang, Zichen Liang, Qian Sun, and Jiaming Zhang. SURGE: On the potential of large language models as general-purpose surrogate code executors. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3268–3308, 2025.

[28] Alberto Maria Metelli, Alessio Russo, and Marcello Restelli. Subgaussian and differentiable importance sampling for off-policy evaluation and learning. *Advances in neural information processing systems*, 34:8119–8132, 2021.

[29] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

[30] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[31] Tom Le Paine, Cosmin Paduraru, Andrea Michi, Caglar Gulcehre, Konrad Zolna, Alexander Novikov, Ziyu Wang, and Nando de Freitas. Hyperparameter selection for offline reinforcement learning. *arXiv preprint arXiv:2007.09055*, 2020.

[32] Doina Precup, Richard S Sutton, and Satinder Singh. Eligibility traces for off-policy policy evaluation. 2000.

[33] Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020.

[34] Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600, 2024.

[35] Leonid Kuvayev Rich Sutton. Model-based reinforcement learning with an approximate, learned model. In *Proceedings of the ninth Yale workshop on adaptive and learning systems*, volume 1996, pages 101–105, 1996.

[36] Cameron Voloshin, Hoang M Le, Nan Jiang, and Yisong Yue. Empirical study of off-policy policy evaluation for reinforcement learning. *arXiv preprint arXiv:1911.06854*, 2019.

[37] Ruoyao Wang, Peter Jansen, Marc-Alexandre Côté, and Prithviraj Ammanabrolu. Scienceworld: Is your agent smarter than a 5th grader? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11279–11298, 2022.

[38] Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fangyuan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, et al. Rag-gym: Optimizing reasoning and search agents with process supervision. *arXiv e-prints*, pages arXiv–2502, 2025.

[39] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380, 2018.

[40] Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. WebShop: Towards scalable real-world web interaction with grounded language agents. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

[41] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*, 2023.

[42] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *Advances in neural information processing systems*, 33:14129–14142, 2020.

[43] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. Combo: Conservative offline model-based policy optimization. *Advances in neural information processing systems*, 34:28954–28967, 2021.

[44] Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*, 2023.

Supplementary Materials

A Limitations

ADWM has two practical scope conditions. First, like all OPE estimators, it is bottlenecked by ground-truth quality: on WebShop the binomial standard error on GT success rate (up to 4.5% with 128 episodes) is comparable to the gap between adjacent LEAP iterates (3–19%), so finer cross-iterate distinctions require substantially more GT episodes. Second, the ψ adapter is tied to the evaluation LLM's embedding space, so each new evaluation LLM requires a short adapter fit, while neither the offline trajectories nor the world model need to be recollected or retrained. Neither condition reflects a fundamental limitation of the underlying formulation.

B Relation to Classical Importance Sampling

We show that the policy-guided target law (2) is the natural generalization of importance sampling to the setting where the behavior policy π_b is unknown, and that it recovers the evaluation trajectory law exactly when $p_\theta = p_{\pi_b}$ and $\alpha = 1$.

Classical importance sampling identity. For any two policies π_b and π_e, the induced trajectory laws satisfy:

$$\begin{aligned} p_{\pi_e}(\tau) &= \mu(o_1) \prod_{t=1}^{T} \pi_e(a_t \mid h_t)\, P(o_{t+1} \mid h_t, a_t) \\ &= \mu(o_1) \prod_{t=1}^{T} \pi_b(a_t \mid h_t)\, P(o_{t+1} \mid h_t, a_t) \cdot \prod_{t=1}^{T} \frac{\pi_e(a_t \mid h_t)}{\pi_b(a_t \mid h_t)} \\ &= p_{\pi_b}(\tau) \cdot \prod_{t=1}^{T} \frac{\pi_e(a_t \mid h_t)}{\pi_b(a_t \mid h_t)}, \end{aligned} \tag{11}$$

where $\mu(o_1)$ and $P(o_{t+1} \mid h_t, a_t)$ denote the initial observation law and the observation kernel induced by μ, P, and $\mathcal{E}$ after marginalizing the latent states, and the environment term $P(o_{t+1} \mid h_t, a_t)$ cancels between numerator and denominator. Taking logarithms,

$$\log p_{\pi_e}(\tau) = \log p_{\pi_b}(\tau) + \sum_{t=1}^{T} \log \pi_e(a_t \mid h_t) - \sum_{t=1}^{T} \log \pi_b(a_t \mid h_t). \tag{12}$$

Equation (12) is the starting point shared by importance sampling, policy-guided diffusion [17], and STITCH-OPE [9]: the target trajectory law differs from the behavior trajectory law by a trajectory-level policy-likelihood correction.

Why π_b is unavailable. In our agentic setting, the offline dataset $\mathcal{D}$ may aggregate trajectories from heterogeneous sources—different prompting strategies, decoding configurations, or agent versions. The behavior policy π_b is therefore a mixture whose density $\pi_b(a_t \mid h_t)$ is generally unavailable in closed form, preventing direct instantiation of (11).

From IS to the policy-guided target law. We replace the unknown $p_{\pi_b}(\tau)$ with a learned generative model $p_\theta(\tau)$ trained on $\mathcal{D}$ by maximum likelihood. Dropping the negative behavior term $-\sum_t \log \pi_b(a_t \mid h_t)$ from (12) acts as a form of behavior regularization [17]: it anchors the target distribution to the support of the offline data rather than allowing unbounded correction away from it. Exponentiating and introducing a strength parameter $\alpha > 0$ yields:

$$q_\alpha(\tau \mid \pi_e) \;\propto\; p_\theta(\tau)\, \exp\Big(\alpha \sum_{t=1}^{T} \log \pi_e(a_t \mid h_t)\Big), \tag{13}$$

which is (2) in the main text.

Recovery of p_{π_e} at $p_\theta = p_{\pi_b}$, $\alpha = 1$. Suppose the learned prior is exact, i.e. $p_\theta = p_{\pi_b}$. Then (13) with $\alpha = 1$ gives:

$$q_1(\tau \mid \pi_e) \;\propto\; p_{\pi_b}(\tau)\, \exp\Big(\sum_{t=1}^{T} \log \pi_e(a_t \mid h_t)\Big)$$
$$= p_{\pi_b}(\tau) \prod_{t=1}^{T} \pi_e(a_t \mid h_t). \tag{14}$$

The normalizing constant of (14) is:

$$Z = \int p_{\pi_b}(\tau) \prod_{t=1}^{T} \pi_e(a_t \mid h_t)\, d\tau$$
$$= \int \mu(o_1) \prod_{t=1}^{T} \pi_b(a_t \mid h_t)\, P(o_{t+1} \mid h_t, a_t) \cdot \prod_{t=1}^{T} \pi_e(a_t \mid h_t)\, d\tau. \tag{15}$$

Using (11) to substitute $p_{\pi_b}(\tau) \prod_t \pi_e(a_t \mid h_t)/\pi_b(a_t \mid h_t) = p_{\pi_e}(\tau)$:

$$Z = \int p_{\pi_b}(\tau) \prod_{t=1}^{T} \frac{\pi_e(a_t \mid h_t)}{\pi_b(a_t \mid h_t)} \cdot \prod_{t=1}^{T} \pi_b(a_t \mid h_t)\, d\tau$$
$$= \int p_{\pi_e}(\tau)\, d\tau \;=\; 1. \tag{16}$$

Therefore (14) is already normalized, and:

$$q_1(\tau \mid \pi_e)\Big|_{p_\theta = p_{\pi_b}} \;=\; p_{\pi_b}(\tau) \prod_{t=1}^{T} \pi_e(a_t \mid h_t) \;=\; p_{\pi_e}(\tau), \tag{17}$$

where the last equality follows from (11). This confirms that q_α with $\alpha = 1$ and $p_\theta = p_{\pi_b}$ recovers the evaluation trajectory law exactly.

Role of α. For $\alpha \neq 1$ or $p_\theta \neq p_{\pi_b}$, $q_\alpha(\tau \mid \pi_e)$ interpolates between the offline prior ($\alpha \to 0$, recovering p_θ) and a distribution that increasingly concentrates on trajectories preferred by π_e ($\alpha \to \infty$). In practice, α is a hyperparameter that balances the fidelity of the simulated environment against the alignment of generated rollouts with the evaluation policy.

C Proof of Proposition 1

We derive the single-step conditional law for o_{t+1} induced by the policy-guided trajectory law $q_\alpha(\tau \mid \pi_e)$ defined in (2).

Fix a history $h_t = (o_1, a_1, \ldots, o_t)$ and a current action a_t. We wish to characterize the conditional distribution $q_\alpha(o_{t+1} \mid h_t, a_t; \pi_e)$. Denote the future suffix trajectory as:

$$\tau_{t+1:T} = (o_{t+1}, a_{t+1}, o_{t+2}, \ldots, a_T, o_{T+1}),$$

so that the full trajectory decomposes as $\tau = (h_t, a_t, \tau_{t+1:T})$.

By definition of q_α in (2), the unnormalized density of any trajectory τ is:

$$q_\alpha(\tau \mid \pi_e) \propto p_\theta(\tau)\, \exp\Big(\alpha \sum_{t'=1}^{T} \log \pi_e(a_{t'} \mid h_{t'})\Big).$$

Conditioning on (h_t, a_t) amounts to fixing all variables up to and including a_t. The terms $\sum_{t'=1}^{t} \log \pi_e(a_{t'} \mid h_{t'})$ in the exponent are therefore constant with respect to the future suffix $\tau_{t+1:T}$, and are absorbed into the normalizing constant. We obtain:

$$q_\alpha(\tau_{t+1:T} \mid h_t, a_t; \pi_e) = \frac{q_\alpha(\tau \mid \pi_e)}{q_\alpha(h_t, a_t \mid \pi_e)}$$
$$\propto p_\theta(\tau_{t+1:T} \mid h_t, a_t)\, \exp\Big(\alpha \sum_{u=t+1}^{T} \log \pi_e(a_u \mid h_u)\Big), \tag{18}$$

where we have used the fact that $p_\theta(\tau) = p_\theta(h_t, a_t) \cdot p_\theta(\tau_{t+1:T} \mid h_t, a_t)$ by the chain rule of probability.

We now factor $p_\theta(\tau_{t+1:T} \mid h_t, a_t)$ by separating out o_{t+1} from the remaining suffix. By the chain rule:

$$p_\theta(\tau_{t+1:T} \mid h_t, a_t) = p_\theta(o_{t+1} \mid h_t, a_t)\, p_\theta(\tau_{t+2:T} \mid h_t, a_t, o_{t+1}), \tag{19}$$

where we write $\tau_{t+2:T} = (a_{t+1}, o_{t+2}, \ldots, a_T, o_{T+1})$ for the suffix starting after o_{t+1}. Note that once o_{t+1} is fixed, the updated history is $h_{t+1} = (h_t, a_t, o_{t+1})$, so $p_\theta(\tau_{t+2:T} \mid h_t, a_t, o_{t+1}) = p_\theta(\tau_{t+2:T} \mid h_{t+1})$.

Substituting (19) into (18):

$$q_\alpha(\tau_{t+1:T} \mid h_t, a_t; \pi_e) \propto p_\theta(o_{t+1} \mid h_t, a_t)\, p_\theta(\tau_{t+2:T} \mid h_{t+1}) \exp\Big(\alpha \sum_{u=t+1}^{T} \log \pi_e(a_u \mid h_u)\Big). \tag{20}$$

To obtain the marginal over o_{t+1}, we integrate (20) over all remaining future variables $\tau_{t+2:T}$:

$$\begin{aligned} q_\alpha(o_{t+1} \mid h_t, a_t; \pi_e) &= \int q_\alpha(\tau_{t+1:T} \mid h_t, a_t; \pi_e)\, d\tau_{t+2:T} \\ &\propto p_\theta(o_{t+1} \mid h_t, a_t) \int p_\theta(\tau_{t+2:T} \mid h_{t+1}) \exp\Big(\alpha \sum_{u=t+1}^{T} \log \pi_e(a_u \mid h_u)\Big)\, d\tau_{t+2:T}. \end{aligned} \tag{21}$$

The integral in (21) can be recognized as an expectation under the prior over future trajectories rooted at o_{t+1}. Specifically, since $p_\theta(\tau_{t+2:T} \mid h_{t+1})$ is a probability distribution over $\tau_{t+2:T}$:

$$\begin{aligned} &\int p_\theta(\tau_{t+2:T} \mid h_{t+1}) \exp\Big(\alpha \sum_{u=t+1}^{T} \log \pi_e(a_u \mid h_u)\Big)\, d\tau_{t+2:T} \\ &= \int p_\theta(\tau_{t+1:T} \mid h_t, a_t, o_{t+1}) \exp\Big(\alpha \sum_{u=t+1}^{T} \log \pi_e(a_u \mid h_u)\Big)\, d\tau_{t+1:T} \\ &= \mathbb{E}_{\tau_{t+1:T} \sim p_\theta(\cdot \mid h_t, a_t, o_{t+1})} \left[\exp\Big(\alpha \sum_{u=t+1}^{T} \log \pi_e(a_u \mid h_u)\Big) \right], \end{aligned} \tag{22}$$

where in the second equality we have re-expressed the integral as an expectation over the full future suffix $\tau_{t+1:T}$ conditioned on (h_t, a_t, o_{t+1}), noting that integrating over $\tau_{t+2:T}$ under $p_\theta(\cdot \mid h_{t+1})$ is equivalent to integrating over $\tau_{t+1:T}$ under $p_\theta(\cdot \mid h_t, a_t, o_{t+1})$ since the o_{t+1} component is fixed in both cases.

The expectation in (22) is precisely the continuation factor $\mathcal{C}_\alpha(o_{t+1}; h_t, a_t, \pi_e)$ defined in (4). Substituting back into (21):

$$q_\alpha(o_{t+1} \mid h_t, a_t; \pi_e) \propto p_\theta(o_{t+1} \mid h_t, a_t)\, \mathcal{C}_\alpha(o_{t+1}; h_t, a_t, \pi_e). \tag{23}$$

This is precisely (3), completing the proof. □

D Proof of Theorem 1

We derive the fused single-step conditional law (5) from Proposition 1.

By Proposition 1, the single-step conditional induced by q_α satisfies:

$$q_\alpha(o_{t+1} \mid h_t, a_t; \pi_e) \propto p_\theta(o_{t+1} \mid h_t, a_t)\, \mathcal{C}_\alpha(o_{t+1}; h_t, a_t, \pi_e). \tag{24}$$

It remains to decompose $p_\theta(o_{t+1} \mid h_t, a_t)$. Consider the joint distribution $p_\theta(o_{t+1}, a_t \mid h_t)$. By the chain rule, this factors in two ways:

$$p_\theta(o_{t+1}, a_t \mid h_t) = p_\theta(o_{t+1} \mid h_t, a_t)\, p_\theta(a_t \mid h_t), \tag{25}$$

$$p_\theta(o_{t+1}, a_t \mid h_t) = p_\theta(a_t \mid o_{t+1}, h_t)\, p_\theta(o_{t+1} \mid h_t). \tag{26}$$

Equating (25) and (26):

$$p_\theta(o_{t+1} \mid h_t, a_t)\, p_\theta(a_t \mid h_t) = p_\theta(a_t \mid o_{t+1}, h_t)\, p_\theta(o_{t+1} \mid h_t). \tag{27}$$

Dividing both sides by $p_\theta(a_t \mid h_t) > 0$:

$$p_\theta(o_{t+1} \mid h_t, a_t) = \frac{p_\theta(a_t \mid o_{t+1}, h_t)\, p_\theta(o_{t+1} \mid h_t)}{p_\theta(a_t \mid h_t)}. \tag{28}$$

Since $p_\theta(a_t \mid h_t)$ is constant with respect to o_{t+1} for fixed (h_t, a_t), (28) gives:

$$p_\theta(o_{t+1} \mid h_t, a_t) \propto p_\theta(o_{t+1} \mid h_t)\, p_\theta(a_t \mid o_{t+1}, h_t). \tag{29}$$

Substituting (29) into (24):

$$q_\alpha(o_{t+1} \mid h_t, a_t; \pi_e) \propto p_\theta(o_{t+1} \mid h_t)\, p_\theta(a_t \mid o_{t+1}, h_t)\, \mathcal{C}_\alpha(o_{t+1}; h_t, a_t, \pi_e), \tag{30}$$

which is the $\omega = \eta = 1$ case of (5).

For general $\omega > 0$ and $\eta \geq 0$, we introduce the steered family by replacing each factor with its ω-th and η-th power respectively. This corresponds to treating ω and η as inverse temperatures on the action posterior and continuation factor, giving:

$$\widetilde{q}_{\omega,\eta}(o_{t+1} \mid h_t, a_t; \pi_e) \propto p_\theta(o_{t+1} \mid h_t)\, p_\theta(a_t \mid o_{t+1}, h_t)^\omega\, \mathcal{C}_\alpha(o_{t+1}; h_t, a_t, \pi_e)^\eta. \tag{31}$$

This is (5), completing the proof. □

Remark 2. *The denominator $p_\theta(a_t \mid h_t)$ in (28) depends only on (h_t, a_t) and not on o_{t+1}. It is therefore a constant with respect to the variable of integration and is absorbed into the normalizing constant of the conditional distribution.*

Remark 3. *The steered family $\widetilde{q}_{\omega,\eta}$ is a valid probability distribution for any $\omega > 0$ and $\eta \geq 0$, provided the normalizing constant*

$$Z(\omega, \eta; h_t, a_t) = \int p_\theta(o_{t+1} \mid h_t)\, p_\theta(a_t \mid o_{t+1}, h_t)^\omega\, \mathcal{C}_\alpha(o_{t+1}; h_t, a_t, \pi_e)^\eta\, do_{t+1}$$

is finite, which holds under standard regularity conditions on p_θ and π_e.

E Derivation of the Guided Score Function

We derive (9) in three steps, using the notation of Section 4.2: (i) decompose the score of the fused conditional into three additive terms; (ii) show that the first two terms are realized by classifier-free guidance; (iii) derive the tractable approximation of the continuation score.

E.1 Score decomposition of the fused conditional

By Theorem 1, the target single-step conditional is:

$$\widetilde{q}_{\omega,\eta}(z_{t+1} \mid h_t^z, a_t; \pi_e) \;\propto\; p_\theta(z_{t+1} \mid h_t^z)^1 \cdot p_\phi(a_t \mid z_{t+1}, h_t^z)^\omega \cdot \mathcal{C}_\alpha(z_{t+1}; h_t^z, a_t, \pi_e)^\eta. \tag{32}$$

Taking logarithms:

$$\log \widetilde{q}_{\omega,\eta}(z_{t+1} \mid h_t^z, a_t; \pi_e) = \log p_\theta(z_{t+1} \mid h_t^z) + \omega \log p_\phi(a_t \mid z_{t+1}, h_t^z) + \eta \log \mathcal{C}_\alpha(z_{t+1}; h_t^z, a_t, \pi_e) + \text{const}, \tag{33}$$

where the constant absorbs the normalizer, which does not depend on z_{t+1}. Differentiating with respect to z_{t+1} gives the exact score:

$$\nabla_{z_{t+1}} \log \widetilde{q}_{\omega,\eta} = \underbrace{\nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z)}_{\text{prior score}} + \omega \underbrace{\nabla_{z_{t+1}} \log p_\phi(a_t \mid z_{t+1}, h_t^z)}_{\text{action-posterior score}} + \eta \underbrace{\nabla_{z_{t+1}} \log \mathcal{C}_\alpha(z_{t+1}; h_t^z, a_t, \pi_e)}_{\text{continuation score}}. \tag{34}$$

Each term has a distinct role: the prior score keeps generated latents on the manifold of the offline data; the action-posterior score steers generation toward latents from which a_t is recoverable; the continuation score steers generation toward latents that open up futures compatible with π_e.

E.2 Realization of the prior and action-posterior scores via classifier-free guidance

We show that the first two terms of (34) are jointly realized by classifier-free guidance [13] without a separate classifier.

Consider the joint density $p_\theta(z_{t+1}, a_t \mid h_t^z)$. By the chain rule it factors in two ways:

$$p_\theta(z_{t+1}, a_t \mid h_t^z) = p_\theta(z_{t+1} \mid h_t^z, a_t)\, p_\theta(a_t \mid h_t^z), \tag{35}$$

$$p_\theta(z_{t+1}, a_t \mid h_t^z) = p_\phi(a_t \mid z_{t+1}, h_t^z)\, p_\theta(z_{t+1} \mid h_t^z). \tag{36}$$

Equating (35) and (36) and dividing both sides by $p_\theta(a_t \mid h_t^z) > 0$:

$$p_\theta(z_{t+1} \mid h_t^z, a_t) = \frac{p_\phi(a_t \mid z_{t+1}, h_t^z)\, p_\theta(z_{t+1} \mid h_t^z)}{p_\theta(a_t \mid h_t^z)}. \tag{37}$$

Taking logarithms and differentiating with respect to z_{t+1}, noting that $p_\theta(a_t \mid h_t^z)$ is constant in z_{t+1}:

$$\nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z, a_t) = \nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z) + \nabla_{z_{t+1}} \log p_\phi(a_t \mid z_{t+1}, h_t^z). \tag{38}$$

Rearranging:

$$\nabla_{z_{t+1}} \log p_\phi(a_t \mid z_{t+1}, h_t^z) = \nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z, a_t) - \nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z). \tag{39}$$

Substituting (39) into the first two terms of (34):

$$\begin{aligned}
&\nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z) + \omega\, \nabla_{z_{t+1}} \log p_\phi(a_t \mid z_{t+1}, h_t^z) \\
&= \nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z) + \omega \Big(\nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z, a_t) - \nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z) \Big) \\
&= (1-\omega)\, \nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z) + \omega\, \nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z, a_t).
\end{aligned} \tag{40}$$

Writing $\omega = 1 + \lambda$, (40) becomes:

$$(1+\lambda)\, \nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z, a_t) - \lambda\, \nabla_{z_{t+1}} \log p_\theta(z_{t+1} \mid h_t^z), \tag{41}$$

which is exactly the classifier-free guidance combination with scale λ. In practice, both scores are approximated by the action-conditioned and unconditional branches of the shared score network s_θ, trained jointly via (8) with action dropout.

E.3 Tractable approximation of the continuation score

The continuation factor is defined as:

$$\mathcal{C}_\alpha(z_{t+1}; h_t^z, a_t, \pi_e) := \mathbb{E}_{\tau_{t+1:T} \sim p_\theta(\cdot \mid h_t^z, a_t, z_{t+1})} \left[\exp\Big(\alpha \sum_{u=t+1}^{T} \log \pi_e(a_u \mid h_u^\pi) \Big) \right]. \tag{42}$$

Computing $\nabla_{z_{t+1}} \log \mathcal{C}_\alpha$ exactly requires differentiating through an expectation over all future trajectories, which is intractable. We derive a tractable approximation via the denoised estimate.

At diffusion step k, the score network provides the posterior mean estimate:

$$\hat{z}_{t+1} = \mathbb{E}[z_{t+1} \mid z_{t+1}^k] = \frac{z_{t+1}^k + (1 - \bar{\alpha}_k)\, s_\theta(z_{t+1}^k, k; h_t^z, a_t)}{\sqrt{\bar{\alpha}_k}}. \tag{43}$$

We substitute $\hat{z}_{t+1}$ for z_{t+1} in (42) and replace the expectation over future trajectories with a single greedy rollout $\hat{\tau}_{t+1:T} = \mathcal{R}_\theta(\hat{z}_{t+1}; h_t^z, a_t)$:

$$\log \mathcal{C}_\alpha(z_{t+1}; h_t^z, a_t, \pi_e) \;\approx\; \alpha \sum_{u=t+1}^{T} \log \pi_e(\hat{a}_u \mid h_u^\pi). \tag{44}$$

Differentiating (44) with respect to $\hat{z}_{t+1}$, and using this gradient in place of the gradient with respect to z_{t+1}^k (that is, dropping the Jacobian of (43)), gives the tractable continuation gradient:

$$\eta\, \nabla_{z_{t+1}} \log \mathcal{C}_\alpha \;\approx\; \eta_k\, \alpha\, \nabla_{\hat{z}_{t+1}} \sum_{u=t+1}^{T} \log \pi_e(\hat{a}_u \mid h_u^\pi), \tag{45}$$

where we write η_k to allow the continuation weight to vary with noise level k, suppressing the signal when $\hat{z}_{t+1}$ is unreliable at high noise and amplifying it late in the reverse process.

E.4 Final guided score

Combining (41) and (45), the complete guided score at noise level k is:

$$s(z_{t+1}^k, k; h_t^z, a_t, \pi_e) = \nabla_{z_{t+1}^k} \log p_\theta(z_{t+1}^k \mid h_t^z) + \omega \nabla_{z_{t+1}^k} \log p_\phi(a_t \mid z_{t+1}^k, h_t^z) + \eta_k \alpha \nabla_{\hat{z}_{t+1}} \sum_{u=t+1}^{T} \log \pi_e(\hat{a}_u \mid h_u^\pi), \tag{46}$$

which is (9) in the main text.

F Experimental Setup Details

F.1 Software stack and reproducibility

All training and OPE rollout code is implemented in PyTorch 2.6 with Transformers 4.56 under Python 3.10. LLM-side π_b/π_e inference is served by vLLM 0.8.5 with bf16 weights and a tensor-parallel size of 1 (one GPU per LLM job). Baselines (FQE, DR, IS, WIS, DM) are implemented to match the COBS reference protocol of Voloshin et al. [36] verbatim except for the per-ε FQE correction described in Appendix G.4. Random seeds $s \in \{0, 1, 2, 3, 4\}$ control (i) the ε-greedy admissible-action draw and (ii) the imagined-rollout DDPM noise sequence. The world model and the ψ adapter are trained *once per behavior pool* with a fixed seed; the five per-cell seeds vary inference-time stochasticity only.

F.2 Hardware and compute budget

All experiments run on a shared cluster node with 2× Intel Xeon Gold 6248R (3.00 GHz, 96 logical cores) and 2× NVIDIA Quadro RTX 8000 (48 GB) GPUs interconnected by PCIe 3.0 x16. We allocate a single GPU per training or evaluation job; multi-GPU parallelism is not used.

Wall-clock budget.

- World-model training (per environment): 50 epochs on 256–744 trajectories at batch size 64 takes 30–90 min.
- ψ-adapter training (per environment): 20 epochs at batch size 8 takes 10–20 min.
- GT collection (per cell, per ε): vLLM-served π_e^ε rolling out for 64–128 episodes takes 10–40 min depending on env horizon.
- ADWM inference (per cell, per seed): 5 ε levels × 32 imagined rollouts × horizon ≤ 30 takes 15–45 min.
- FQE / DR (per cell, per seed): 5 ε-conditioned Q-nets at 20 epochs each takes 5–10 min.

Total cost. The full camera-ready experimental suite (six (π_b, π_e) cells × 5 seeds, plus 7 ablation variants on three environments, plus 5 baseline estimators per cell) consumes approximately 300 GPU-hours, of which $\sim$70% is LLM-side (π_b and π_e) inference and $\sim$25% is GT collection. World-model training itself accounts for $< 5\%$ of the budget — the dominant cost of ADWM is the same as the dominant cost of any LLM-agent OPE method, namely running the eval LLM.

F.3 Behavior-policy data

For each (π_b, π_e) cell we collect a single offline behavior pool that is reused across all ε levels and seeds for that cell. The world model is trained *only* on this pool. Table 3 summarises the per-cell trajectory counts, behavior models actually used, and observed empirical statistics. The HotpotQA, ScienceWorld, and ALFWorld-iter1 cells use a single behavior LLM each. The ALFWorld-iter3 and WebShop-iter1 cells use a three-LLM behavior pool consisting of the generic Llama-3.1-8B-Instruct together with two LEAP iterates (iter$_0$ and iter$_2$); we exclude iter$_1$ from the WS pool and iter$_3$ from the ALF pool to ensure the eval policy is never in the behavior pool. This pooling choice mirrors a realistic deployment in which multiple released checkpoints from the same LEAP self-improvement

Table 3: Behavior pool composition and statistics, per cell. "Avg. horizon" is the mean number of environment steps per π_b trajectory; "π_b succ. rate" is the empirical success rate on the collected pool.

Cell	π_b pool	N traj.	Avg. horizon	π_b succ. rate
HotpotQA-DPO/PRM/cross	ReAct-HotpotQA-SFT (single)	512	6.4	12.5%
ScienceWorld-ETO	sw-llama-sft (single)	512	9.1	11.7%
ALFWorld-iter1	Llama-3.1-8B-Instruct (single)	256	29.5	3.5%
ALFWorld-iter3	Llama-3.1-8B-Instr. + leap-alf-iter{0,2}	744	29.2	5.8%
WebShop-iter1	Llama-3.1-8B-Instr. + leap-ws-iter{0,2}	640	14.4	5.0%

chain may already be available offline. We verify post-hoc that no training trajectory in the pool was generated by the exact π_e checkpoint scored at evaluation time.

Trajectories are collected with greedy LLM decoding (temperature 0) and two system-prompt variants $\{p_0, p_1\}$ that differ only in instruction phrasing (not task content), to obtain useful behavior coverage without changing the underlying π_b. Episode length is capped at 30 environment steps for ALFWorld and WebShop and at 12 for HotpotQA and ScienceWorld. Each recorded trajectory contains, per step,

$$(\text{obs_text}_t, \text{llm_response}_t, u_t, a_t, a^t_{\text{valid}}, r_t, \text{done}_t),$$

where u_t is the raw action text parsed from the LLM response and a_t is the action actually passed to the environment after the environment-specific projection interface. We write this interface as

$$a_t \sim \Gamma_{\text{env}}(\cdot \mid h_t, u_t, \mathcal{A}_{\text{adm}}(h_t)) .$$

The world-model transition tuples use the executed action a_t, so (o_{t+1}, r_t) is paired with the action that generated it. The raw response and u_t are retained as auxiliary logged text. Successful trajectories are upsampled $5\times$ during world-model training to counter class imbalance on the long-tail success distributions of Table 3.

F.4 Ground-truth episode counts

Per-cell ground-truth success curves are obtained by running π_e^ε in the actual environment for n_{GT} independent episodes per ε level, with seed 42 fixed across ε within a policy:

Cell	n_{GT} per ε
HotpotQA-DPO	128
HotpotQA-PRM	64
ScienceWorld-ETO	64
ALFWorld-iter1 / iter3	72
WebShop-iter1	128

The binomial standard error on the GT success rate at n=64 is $\leq 6.3\%$ for any $p \in (0, 1)$, and $\leq 4.5\%$ at n=128.

F.5 ε-greedy mixing protocol

Let $\widetilde{\pi}_e(u \mid h)$ denote the evaluation LLM's distribution over raw action text and let $\Gamma_{\text{env}}(a \mid h, u, \mathcal{A}_{\text{adm}}(h))$ denote the environment-specific projection from raw text to an executed admissible action. The induced executed-action policy is

$$\pi_{e,\Gamma}(a \mid h) = \sum_u \widetilde{\pi}_e(u \mid h)\, \Gamma_{\text{env}}(a \mid h, u, \mathcal{A}_{\text{adm}}(h)) . \tag{47}$$

This notation also covers a deterministic projection, in which Γ_{env} is a point mass.

We apply ε-mixing to the executed-action policy:

$$\pi_e^\varepsilon(a \mid h) = (1-\varepsilon)\pi_{e,\Gamma}(a \mid h) + \varepsilon \frac{\mathbb{I}[a \in \mathcal{A}_{\text{adm}}(h)]}{|\mathcal{A}_{\text{adm}}(h)|}. \tag{48}$$

Operationally, we sample one Bernoulli variable per step. With probability $1-\varepsilon$, the LLM produces $u_t \sim \widetilde{\pi}_e(\cdot \mid h_t)$ and the interface samples $a_t \sim \Gamma_{\text{env}}(\cdot \mid h_t, u_t, \mathcal{A}_{\text{adm}}(h_t))$. With probability ε, we sample a_t uniformly from the current admissible set. Both ADWM imagined rollouts and ground-truth environment rollouts use this same interface and random seed.

F.6 World-model architecture

The world model is a 7.4M-parameter latent diffusion model with the modular structure shown in Table 4. In this appendix we write f_θ, $\bar{h}_t$, ϵ_θ, R_θ, and D_θ for the encoder E, the history summary of h_t^z, the noise-prediction form of the score network s_θ, the reward head r_ρ, and the termination head d_ρ of Section 4.2. The largest component is the text-based observation encoder (5.05 M params), which tokenises raw obs_text$_t$ via a stable hash-bucket tokeniser (vocabulary size 32,768, max length 128 tokens) and embeds it into the latent dimension d_z=64 through a 4-layer TransformerEncoder of width 128 with 4 attention heads. The history encoder is a causal 4-layer Transformer with width 128 and a generated subsequent-token mask, conditioning on (z_t, a_t) tuples. The score network is a 3-layer FiLM-MLP of width 256 with sinusoidal time embedding (dimension 64) and zero-initialised output projection; the IDM and BC heads are 2-layer MLPs over (z, h). All non-linearities are SiLU except the IDM / policy / projector heads, which use ReLU and GELU respectively.

Table 4: Per-module parameter counts in the ADWM world model (7.38M parameters total for d_z=64).

Module	Function	Params (M)	% of total
Observation encoder (f_θ)	Text $\to z \in \mathbb{R}^{d_z}$	5.053	68.4%
History encoder ($\bar{h}$)	Causal Transformer over (z, a) pairs	0.814	11.0%
Soft-token projector	$z \to \mathbb{R}^{n_{\text{soft}} \times d_{\text{llm}}}$ for ψ adapter	0.609	8.2%
Score net (ϵ_θ)	FiLM-MLP for diffusion denoising	0.527	7.1%
BC policy head	$z, h \to a$ logits	0.170	2.3%
Reward head (R_θ)	2-layer MLP $\to \sigma(\cdot)$	0.115	1.6%
Termination head (D_θ)	2-layer MLP $\to$ BCE logit	0.050	0.7%
Inverse-dynamics head (IDM)	$z_t, z_{t+1}, h \to a_t$	0.044	0.6%
Action embedding	$\mathbb{Z} \to \mathbb{R}^{d_a}$, $\lvert\mathcal{A}\rvert = 256$, d_a=32	0.001	$<0.1\%$

The action vocabulary $|\mathcal{A}|$ is set per environment from the unique projected actions a_t actually executed in the behavior pool, capped at 256; HotpotQA uses a fixed 4-action vocabulary $\{\langle\text{unk}\rangle, \text{search}, \text{lookup}, \text{finish}\}$ that matches the ReAct protocol of Yao et al. [41].

F.7 World-model training objective

The world model is trained with the multi-task objective

$$\mathcal{L}_{\text{WM}} \;=\; \mathcal{L}_{\text{DSM}} + \lambda_{\text{idm}}\,\mathcal{L}_{\text{IDM}} + \lambda_{\text{bc}}\,\mathcal{L}_{\text{BC}} + \lambda_{\text{div}}\,\mathcal{L}_{\text{div}} + \lambda_{\text{r}}\,\mathcal{L}_{\text{r}} + \lambda_{\text{d}}\,\mathcal{L}_{\text{d}}, \tag{49}$$

with weights $(\lambda_{\text{idm}}, \lambda_{\text{bc}}, \lambda_{\text{div}}, \lambda_{\text{r}}, \lambda_{\text{d}}) = (0.1,\ 0.1,\ 0.05,\ 10,\ 1)$. The component losses are defined as follows.

(a) Denoising score matching ($\mathcal{L}_{\text{DSM}}$). For a sampled diffusion step $k \sim \text{Unif}(1, K)$, latent $z \sim f_\theta(\text{obs}_{t+1})$, history $\bar{h}_t$, and action a_t, we form the noised latent $z_k = \sqrt{\bar{\alpha}_k}\, z + \sqrt{1-\bar{\alpha}_k}\,\xi$ with $\xi \sim \mathcal{N}(0, I)$ and a cosine $\bar{\alpha}_k$ schedule. We train the conditional score network $\epsilon_\theta(z_k, k, \bar{h}_t, a_t)$ together with its unconditional counterpart $\epsilon_\theta(z_k, k, \bar{h}_t, \varnothing)$ under classifier-free dropout p_{uncond}=0.1:

$$\mathcal{L}_{\text{DSM}} \;=\; \mathbb{E}_{k,\xi}\big[w(k)\,\big(\|\epsilon_\theta(z_k, k, \bar{h}_t, a_t) - \xi\|_2^2 + \|\epsilon_\theta(z_k, k, \bar{h}_t, \varnothing) - \xi\|_2^2\big)\big], \quad w(k) \;=\; 1 - \bar{\alpha}_k. \tag{50}$$

The $w(k) = 1 - \bar{\alpha}_k$ weighting keeps the per-step loss numerically comparable across k. This is the noise-prediction form of (8), with $s_\theta = -\epsilon_\theta / \sqrt{1-\bar{\alpha}_k}$.

(b) Inverse-dynamics regulariser ($\mathcal{L}_{\text{IDM}}$). A discriminative auxiliary that predicts the bridging action between two consecutive latents: $\mathcal{L}_{\text{IDM}} = \text{CE}\big(\text{IDM}(z_{t+1}, \bar{h}_t),\, a_t\big)$. This pushes z to be *action-aware* and prevents the diffusion prior from collapsing onto policy-invariant features.

(c) Behavior-cloning soft-token loss ($\mathcal{L}_{\text{BC}}$). $\mathcal{L}_{\text{BC}} = \text{CE}\big(\text{BC}(z_t, \bar{h}_t), a_t\big)$. Trains the latent-to-policy interface against logged actions *independently* of any specific evaluation policy, providing the ψ-adapter retrieval target a stable signal.

(d) Diversity regulariser ($\mathcal{L}_{\text{div}}$). A cross-trajectory contrastive penalty that prevents the latent space from collapsing across distinct trajectories. For batch $\{z^{(i)}_{1:L}\}_{i=1}^{B}$, let $\bar{z}^{(i)} = \text{normalize}(\text{mean}_t z^{(i)}_t)$ be the per-trajectory pooled latent. Then

$$\mathcal{L}_{\text{div}} \;=\; \frac{1}{B(B-1)} \sum_{i \neq j} \big|\bar{z}^{(i)\top} \bar{z}^{(j)}\big|. \tag{51}$$

Same-window adjacent latents are deliberately left free since the dynamics constraint requires them to stay close.

(e) Reward head ($\mathcal{L}_{\text{r}}$). For environments with continuous reward (HotpotQA F1, ScienceWorld shaped reward, WebShop product match) we use MSE on sigmoid-clipped reward-to-go targets: $\mathcal{L}_{\text{r}} = \mathbb{E}_t\big[(\sigma(R_\theta(z_{t+1}, \bar{h}_t)) - r^{\text{tgt}}_t)^2\big]$ with $r^{\text{tgt}}_t \in [0, 1]$. For sparse-reward ALFWorld we replace the MSE by a balanced binary cross-entropy with sigmoid output and per-batch positive-class weight $\min(50, n_{\text{neg}}/n_{\text{pos}})$, which prevents the reward head from collapsing to the constant $p \approx \bar{r}$ predictor. The reward head is supervised on *terminal steps only* when the environment provides a terminal-only success label, and on all steps when shaped per-step rewards are available.

(f) Termination head ($\mathcal{L}_{\text{d}}$). A binary cross-entropy with class-balanced positive weight $\min(50, n_{\text{neg}}/n_{\text{pos}})$: $\mathcal{L}_{\text{d}} = \text{BCE}\big(D_\theta(z_{t+1}, \bar{h}_t),\, \mathbf{1}[\text{done at } t+1]\big)$. Used during imagined rollouts to terminate trajectories that the world model predicts have ended.

F.8 World-model training procedure

Algorithm 1 summarises the optimization . Each forward pass sampling a window of length L within a trajectory builds a rolling-window prediction target: given $(z_{1:L}, a_{1:L-1})$, the model is trained to score the noisy version of every z_{t+1} conditional on $\bar{h}_t$ and a_t. The optimiser is AdamW with $(\beta_1, \beta_2) = (0.9, 0.999)$, weight decay 10^{-4}, peak learning rate 3×10^{-4}, cosine annealing to 10% of peak over 50 epochs, batch size 64, gradient norm clip 1.0. WebShop and ALFWorld train under PyTorch automatic mixed precision (bf16 master + fp32 accumulator); HotpotQA and ScienceWorld train in fp32 because the smaller dataset fits comfortably in 48 GB.

Algorithm 1 World-model training (per environment).

Require: Behavior pool $\mathcal{D}_b = \{\tau_i\}$, window length L, total epochs E
Initialise WM parameters θ
for epoch $e = 1, \ldots, E$ **do**
  for minibatch $\mathcal{B} \subset \mathcal{D}_b$ of size B **do**
    Sample window $(o_{t:t+L},\, a_{t:t+L-1},\, r_{t+1:t+L},\, \text{done}_{t+1:t+L})$ from each $\tau \in \mathcal{B}$
    Encode latents $z_{t:t+L} \leftarrow f_\theta(o_{t:t+L})$
    Build histories $\bar{h}_t \leftarrow \text{HistEnc}(z_{1:t}, a_{1:t-1})$
    Compute $\mathcal{L}_{\text{WM}}$ via Eq. 49
    Update θ with an AdamW step on $\mathcal{L}_{\text{WM}}$
    Apply gradient clipping at $\|\nabla\| = 1.0$
  end for
  Step cosine LR schedule
end for
Save best checkpoint by held-out reward MSE.

F.9 ψ-adapter training

The ψ adapter projects latent observations into a sequence of soft tokens that drop into the input embedding space of π_e. This step is necessary because Eq. 9's continuation guidance backpropagates a gradient *through* π_e's frozen input layer; without an adapter that targets π_e's embedding manifold,

the gradient is meaningless. The adapter $G_\psi : \mathbb{R}^{d_z} \to \mathbb{R}^{n_{\text{soft}} \times d_{\text{llm}}}$ is a 3-layer GELU MLP ($d_z \to 256 \to 256 \to 8 \cdot d_{\text{llm}}$) with $n_{\text{soft}} = 8$ soft tokens.

The training objective is a symmetric InfoNCE term plus a $0.1\times$ MSE regulariser:

$$\mathcal{L}_\psi = \underbrace{-\frac{1}{2B}\sum_{i=1}^{B}\Big(\log\frac{e^{s_{ii}/\kappa}}{\sum_j e^{s_{ij}/\kappa}} + \log\frac{e^{s_{ii}/\kappa}}{\sum_j e^{s_{ji}/\kappa}}\Big)}_{\mathcal{L}_{\text{InfoNCE}}} + 0.1 \cdot \underbrace{\frac{1}{B}\sum_{i=1}^{B}\big\|\bar{G}_\psi(z_i) - \bar{e}_i\big\|_2^2}_{\mathcal{L}_{\text{MSE}}}, \tag{52}$$

where $s_{ij} = \text{cossim}(\bar{G}_\psi(z_i), \bar{e}_j)$, $\bar{G}_\psi(z_i) = \text{mean}_n G_\psi(z_i)_n$ is the soft-token-mean of the projector, $\bar{e}_j$ is the LLM-token-mean of the ground-truth π_e input embedding sequence, and κ=0.1. Pure MSE ($\mathcal{L}_{\text{InfoNCE}} \to 0$) collapses because $\bar{e}_j$ has low variance across observations, yielding $\partial G_\psi / \partial z \to 0$ and a vacuous Eq. 9 gradient; pure InfoNCE produces a discriminative adapter whose absolute outputs drift out-of-distribution for the LLM. The combined loss balances both. We optimise with Adam (lr=10^{-4}), batch size 8, for 20 epochs.

Final-epoch top-1 retrieval accuracy on the InfoNCE batch (random baseline $1/B = 0.125$):

Cell	ψ top-1
HotpotQA	0.715
ScienceWorld	0.394
ALFWorld-iter1	0.558
ALFWorld-iter3	0.470
WebShop	$\sim$ 0.10

The WebShop adapter sits at the random-baseline floor: token-mean target embeddings of long product descriptions are nearly degenerate under cosine similarity, so the InfoNCE objective alone cannot discriminate them at batch size 8. Despite this, the adapter still provides a useful Eq. 9 *gradient* (rather than a useful retrieval) on WebShop, which is why ablating it still degrades ρ by 0.30 (Fig. 4).

F.10 OPE-rollout protocol

Algorithm 2 summarises the imagined-rollout procedure used to compute $\hat{J}$ at evaluation time. Each rollout is initialised from a uniformly sampled real-environment obs_0 in the behavior pool and encoded into $z_0 \sim f_\theta(\text{obs}_0)$. At each step, the evaluation LLM first produces raw action text u_t from the soft-token state $G_\psi(z_t)$. The same environment-specific projection used during data collection then maps u_t to the executed action a_t, with the ε-random branch applied as in Eq. 48. The diffusion world model samples z_{t+1} conditioned on this executed action a_t, using both Eq. 8 (local CFG) and Eq. 9 (continuation guidance) at each denoising step. We use n_{roll}=32 rollouts per cell (n_{roll}=64 on ALFWorld-iter1 to compensate for binary-success Monte Carlo noise), maximum horizon $H_{\max}$=30 steps, n_{denoise}=50 DDPM denoising steps per environment step, λ_{CFG}=1.0, and continuation-guidance hyperparameters (α=1.0, $\eta_{\max}$=0.5, h=5). Rollouts terminate early when the termination head $D_\theta(z_t, h_t)$ predicts $\Pr[\text{done}] \geq 0.5$. Empirical mean rollout lengths after early termination: HP 8, SW/WS/ALF-iter3 12, ALF-iter1 18.

F.11 Example interaction trace: WebShop

To make the interaction notation in Section 3.1 concrete, we present two representative logged WebShop episodes from the 640-trajectory behavior pool used for the WebShop-iter1 cell in Table 3, comprising Llama-3.1-8B-Instruct and `leap-ws-iter{0,2}` trajectories. Direct inspection of the pool gives $N = 640$, mean horizon 14.42, and behavior-policy success rate 5.0%, matching Table 3. Both examples below were generated by the Llama-3.1-8B-Instruct component, with episode identifiers 18 and 17. Thus, the logging policy and data source are consistent with the pool used to train the WebShop world model.

A logged episode has the form

$$\tau = (o_1, a_1, r_1, \ldots, o_T, a_T, r_T, o_{T+1}).$$

At step t, the policy conditions on the full interaction history

$$h_t = (o_1, a_1, \ldots, o_{t-1}, a_{t-1}, o_t)$$

Algorithm 2 ADWM imagined rollout for OPE.

Require: Trained $f_\theta, \bar{h}, \epsilon_\theta, R_\theta, D_\theta, G_\psi$; evaluation LLM $\widetilde{\pi}_e$; action interface Γ_{env}; admissible-action function $\mathcal{A}_{\text{adm}}$; success bank $\mathcal{B}_\succ$

Require: $n_{\text{roll}}, H_{\max}, n_{\text{denoise}}, \lambda_{\text{CFG}}, \alpha, \eta_{\max}, h, \varepsilon$

for $i = 1, \ldots, n_{\text{roll}}$ **do**
  Sample obs_0 uniformly from behavior pool; $z_0 \leftarrow f_\theta(\text{obs}_0)$
  for $t = 0, \ldots, H_{\max} - 1$ **do**
    Project to soft tokens $\tilde{s}_t \leftarrow G_\psi(z_t)$
    Sample $b_t \sim \text{Bernoulli}(\varepsilon)$
    if $b_t = 1$ **then**
      Sample $a_t \sim \text{Unif}(\mathcal{A}_{\text{adm}}(h_t))$
    else
      Sample raw action text $u_t \sim \widetilde{\pi}_e(\cdot \mid \tilde{s}_t, \text{prompt})$
      Sample executed action $a_t \sim \Gamma_{\text{env}}(\cdot \mid h_t, u_t, \mathcal{A}_{\text{adm}}(h_t))$
    end if
    Initialise $z_{t+1}^{(K)} \sim \mathcal{N}(0, I)$
    for $k = K, \ldots, 1$ **do**
      $\hat{\epsilon} \leftarrow (1 + \lambda_{\text{CFG}})\epsilon_\theta(z^{(k)}, k, \bar{h}_t, a_t) - \lambda_{\text{CFG}}\epsilon_\theta(z^{(k)}, k, \bar{h}_t, \varnothing)$
      $g_k \leftarrow \alpha\, \eta(k; \eta_{\max}) \nabla_{\hat{z}} \sum_{u=t+1}^{t+h} \log \widetilde{\pi}_e(\hat{a}_u \mid G_\psi(\hat{z}_u))$, with $\hat{z}$ the denoised estimate and $\hat{z}_u$ an h-step world-model rollout from it
      $z^{(k-1)} \leftarrow \text{DDPM_step}(z^{(k)}, \hat{\epsilon} - g_k)$
    end for
    $z_{t+1} \leftarrow z^{(0)}$; $\bar{h}_{t+1} \leftarrow \text{HistEnc}(z_{0:t+1}, a_{0:t})$
    Record $\cos_t \leftarrow \cos(z_{t+1}, \mathcal{B}_\succ)$, $r_t \leftarrow \sigma(R_\theta(z_{t+1}, \bar{h}_t))$
    if $\sigma(D_\theta(z_{t+1}, \bar{h}_t)) \geq 0.5$ **then**
      break
    end if
  end for
  Record $\hat{J}^{(i)} \leftarrow \max_t \cos_t$
end for
return $\hat{J} \leftarrow \frac{1}{n_{\text{roll}}} \sum_i \hat{J}^{(i)}$

and produces a text action a_t, stored as a reasoning response followed by an `<action>...</action>` tag. WebShop then returns the next observation o_{t+1}, reward $r_t \in [0, 1]$, and a termination indicator. For Episode 1, we report the complete action–reward sequence and expand three representative reasoning steps. Episode 2 is shown as a concise trace focused on the graded terminal reward.

Episode 1: realistic long-horizon trace (episode 18, $T = 16$)

This episode is close to the pool mean horizon of 14.42. We retain its malformed, repeated, and internally inconsistent actions because they expose behavior-policy failure modes that a short, clean trajectory would hide.

Task instruction. *Find me easy apply, cruelty free, long lasting makeup palettes, and price lower than $40.00 dollars.*

Complete action–reward sequence. The note column describes only properties directly visible in the logged action string or navigation sequence; it does not introduce additional training labels.

t	Action a_t	r_t	Logged behavior
1	`search[easy apply, cruelty free, long lasting makeup palettes, price lower than 40.00 dollars]`	0	Concrete search query.
2	`click[back to search]`	0	Returns to the search page.
3	`search[<query>]`	0	Emits the literal template placeholder.
4	`click[back to search]`	0	Returns to the search page again.
5	`search[<query>] and click[search]`	0	Concatenates two action expressions into one string.
6	`search[easy apply, cruelty free, long lasting makeup palettes under 40.00 dollars]`	0	Issues a concrete reformulated query.
7	`click[b07cxtnvrj]`	0	Opens the first product candidate.
8	`click[back to search]`	0	Returns to the result list.
9	`search[easy apply, cruelty free, long lasting makeup palettes under 40.00 dollars]`	0	Repeats the query from step 6.
10	`click[next >]`	0	Moves to the next result page.
11	`click[b08xqrddt1]`	0	Opens a second product candidate.
12	`click[set-a]`	0	Selects one product option.
13	`click[features]`	0	Opens the feature panel.
14	`click[< prev]`	0	Returns to the product page.
15	`click[set-b]`	0	Switches to another product option.
16	`click[buy now]`	**1.0**	Purchases and terminates the episode.

Step 1: initial search. The initial observation is

$$o_1 = \textit{“WebShop [SEP] Instruction: [SEP] \{task above\} [SEP] Search”}, \qquad h_1 = (o_1).$$

The full logged reasoning is:

> *“Since the task is to find easy apply, cruelty free, long lasting makeup palettes, and the price is lower than 40.00 dollars, I should look for makeup palettes that meet these criteria. I will search through the available products and filter them based on the specified requirements.”*

Step 5: malformed composite action. By step 5, the history contains two failed search attempts, including the literal placeholder `search[<query>]` at step 3. The full logged reasoning is:

> *“My current observation is WebShop [SEP] Instruction: [SEP] {task} [SEP] Search. The admissible actions for this situation are search[<query>] and click[search]. I will choose the action that best advances the shopping goal.”*

The emitted string is nevertheless `search[<query>] and click[search]`, which concatenates two action expressions instead of producing one admissible action. This discrepancy is present in the trajectory log and is not introduced by our transcription.

Step 16: reasoning–action disagreement. The final observation o_{16} is a product page with `set-a`/`set-b` options, reached after the policy switches between the two sets at steps 12 and 15. The full logged reasoning is:

> *“Since the current goal is to find an easy-to-apply, cruelty-free, long-lasting makeup palettes. The current observation is that the current page is showing results for makeup palettes, but the results do not match the current goal. [. . .] I will look at the list of admissible actions to find a matching action.”*

Despite explicitly stating that the product does not match, the policy emits `click[buy now]`. WebShop assigns the resulting purchase $r_{16} = 1.0$. Thus, the policy's verbalized belief, its executed action, and the environment outcome need not agree. OPE must predict the realized action-conditioned outcome rather than infer reward from the policy's stated confidence.

Reconstructed post-terminal display. Our data collector discards the raw post-terminal page after recording $\mathrm{done} = 1$, so o_{17} is not a logged trajectory field. WebShop's `done_page.html` template displays *“Thank you for shopping with us!”* together with the numerical score. We report this template-derived wording only for completeness.

Episode 2: graded partial reward (episode 17, $T = 5$)

Task instruction. *Find me wash cold, machine wash men's shirts with polyester heathers, heathers cotton, cotton heather, needle sleeve, classic fit with color: purple, and fit type: men, and size: x-small, and price lower than $40.00 dollars.*

After searching and opening product `b09rvf4jp1`, the policy reaches an options page offering

fit type: `men`, `women`,
color: `black`, `navy`, `dark heather`, `neon pink`, `royal blue`,
size: `small`, `medium`, `large`, `x-large`, `xx-large`.

The requested color `purple` and size `x-small` do not appear among the options displayed for this listing. The policy selects `click[men]`, matching the available fit-type option, and then executes `click[buy now]`. The recorded terminal reward is

$$r_5 = 0.727\overline{272},$$

rather than 0 or 1.
WebShop computes a graded match score using product type, requested attributes, selected options, and the price constraint:

$$r = r_{\text{type}} \frac{n_{\text{attr}} + n_{\text{option}} + r_{\text{price}}}{|\mathcal{A}_{\text{goal}}| + |\mathcal{O}_{\text{goal}}| + 1}.$$

The observed reward therefore reflects a partial match. Because our logging schema does not retain the environment's internal goal object or the component-wise reward breakdown, we do not attribute the decimal score to a unique set of matched and unmatched terms. The trace supports only the directly observable conclusion that the selected listing lacks the requested color and size options and nevertheless receives substantial partial credit.

Episode	Steps T	Terminal r_T	Main phenomenon
1: makeup palette	16	1.0	The purchase succeeds despite the policy's reasoning stating that the product does not match.
2: men's shirt	5	0.727	The selected listing does not offer the requested color or size, but receives graded partial credit.

Together, the episodes illustrate three properties of the logged WebShop process that matter for OPE. First, behavior-policy outputs can contain literal placeholders, repeated searches, and concatenated actions. Second, the policy's reasoning text may disagree with its executed action and with the resulting environment reward. Third, WebShop returns multi-valued terminal rewards, whereas ALFWorld uses sparse binary success.

Graded rewards do not themselves cause the variance of IS-family estimators; that instability is driven primarily by products of behavior-to-evaluation policy ratios over long trajectories.

We selected Episode 2 for its short, directly verifiable partial-reward contrast. Episode 1 was selected because its horizon is close to the behavior pool average and because it exposes realistic logging and policy failures, rather than because it is a clean or unusually favorable trajectory. Both examples follow the same observation–action–reward schema used throughout the WebShop behavior pool.

F.12 Verified action-projection trace: ALFWorld

The WebShop examples above illustrate irregularities in an agent's own outputs. ALFWorld exposes a complementary issue: the action stored in a trajectory can differ from the action text produced by the LLM because the collection pipeline projects malformed or inadmissible outputs onto the current admissible-command set. We verified the following trace against the exact step-level log, including its 29 recorded `admissible_commands`, and against the projection and repair logic of our data collection pipeline.

Deterministic action repair at step $t = 6$

Raw LLM output u_6. At step 6, the policy produces

```
REASON:
I need to move the apple to the garbage can.
```

```
ACTION:
go to garbagecan 1.
```

The trailing period is part of the model output.

Projected and executed action a_6. The action interface maps the raw output to the action stored in the trajectory and passed to ALFWorld:

$$a_6 = \texttt{go to cabinet 1.}$$

Why the two strings differ. The discrepancy is produced by the following deterministic code path.

1. The projection function extracts `go to garbagecan 1.` from the `ACTION:` block and tests exact, case-normalized membership in the current admissible-command pool.
2. The exact step log contains 29 admissible commands, but does not contain `go to garbagecan 1`. The agent had already reached `garbagecan 1` at the preceding step, and ALFWorld does not offer an action that navigates to the current location. The task-progressing command available in this state is `move apple 2 to garbagecan 1`, which the policy eventually executes at step 8.
3. Because exact matching fails, the collector first attempts its prefix-matching repair. No candidate is found for the extracted string `go to garbagecan 1.`.
4. The collector therefore applies token-overlap scoring. The extracted token set is

   $$\{\texttt{go}, \texttt{to}, \texttt{garbagecan}, \texttt{1.}\}.$$

   The punctuation-attached token `1.` does not match the pool token `1`. Consequently, `go to cabinet 1` shares $\{\texttt{go}, \texttt{to}\}$, while `move apple 2 to garbagecan 1` shares $\{\texttt{to}, \texttt{garbagecan}\}$. Both receive overlap score 2.
5. The scoring loop updates the current best action only under a strict > comparison. Equal-scoring candidates therefore do not replace the first maximal entry encountered in admissible-list order. `go to cabinet 1` is the first command attaining score 2, and it appears before `move apple 2 to garbagecan 1`. It is therefore retained as the repaired action.

Candidate command	Shared tokens	Score
`go to cabinet 1`	$\{\texttt{go}, \texttt{to}\}$	2
`move apple 2 to garbagecan 1`	$\{\texttt{to}, \texttt{garbagecan}\}$	2

No random branch is invoked at this step. Random choice is used only in the collector's prefix-candidate branch or its final fallback branch, neither of which determines this action. Given the raw model string and the logged admissible-command order, the substitution is reproducible.

Step	Executed action	Effect
6	`go to cabinet 1`	Deterministic repair redirects the agent away from the garbage can.
7	`go to garbagecan 1`	The policy navigates back to the intended receptacle.
8	`move apple 2 to garbagecan 1`	The policy executes the task-progressing command and completes the task.

This trace clarifies which action an offline evaluator must model. The raw reasoning and `ACTION:` string define u_6 and explain the policy's intent, but the transition is generated by the projected action a_6 stored in the trajectory. Thus, the relevant conditional is

$$P(o_7, r_6 \mid h_6,\ a_6 = \texttt{go to cabinet 1}),$$

not a transition conditioned on the unexecuted raw string `go to garbagecan 1.`.

Our world-model training tuples use the projected action a_t that was actually passed to the environment. The raw LLM response and u_t are retained only as auxiliary logged text. Consequently, (o_{t+1}, r_t) is aligned with the executed action that generated it; the discrepancy illustrated here is not a transition-label error in ADWM's training data. The same projection interface is applied during imagined evaluation-policy rollouts before the world model predicts the next latent state.

The trailing period contributes to the failed string normalization and to the token-overlap tie, but the more fundamental trigger is that the model proposes an action that is not admissible in the current state. The resulting detour is therefore neither environment stochasticity nor random replacement; it is a deterministic side effect of the collector's admissible-action repair rule.

G Baseline Implementation Details

All baselines follow the COBS reference protocol of Voloshin et al. [36] unless stated otherwise. The full set of estimators is included in our released code.

G.1 Behavior log-probability extraction

For likelihood-based baselines, the distinction between raw LLM output u_t and projected action a_t matters. The exact probability of an executed action under an LLM-plus-projection system is

$$\pi_\Gamma(a_t \mid h_t) = \sum_u \widetilde{\pi}(u \mid h_t)\, \Gamma_{\text{env}}(a_t \mid h_t, u, \mathcal{A}_{\text{adm}}(h_t))\,, \tag{53}$$

which aggregates every raw output that the projection maps to a_t. Enumerating this preimage is intractable for an open-vocabulary LLM.

Following the COBS-style implementation used by our likelihood baselines, we therefore score the canonical logged action string a_t under the behavior and evaluation LLMs. We re-run each LLM in inference mode and extract token log-probabilities over that action span. Longest-common-token-prefix matching handles BPE merge boundaries between the prompt and action region. Let $\bar{\pi}_b(a_t \mid h_t)$ and $\bar{\pi}_e(a_t \mid h_t)$ denote these token-likelihood proxies. We apply ε-mixing at the probability level,

$$\bar{\pi}_e^{\varepsilon}(a_t \mid h_t) = (1-\varepsilon)\bar{\pi}_e(a_t \mid h_t) + \varepsilon \frac{\mathbb{I}[a_t \in \mathcal{A}_{\text{adm}}(h_t)]}{|\mathcal{A}_{\text{adm}}(h_t)|},$$

and use

$$\rho_t = \frac{\bar{\pi}_e^{\varepsilon}(a_t \mid h_t)}{\bar{\pi}_b(a_t \mid h_t)}$$

in IS, WIS, and DR.

Because the behavior data are generated with greedy decoding, these token likelihoods are also proxies for the probabilities of the deterministic behavior policy. These are proxy ratios over canonical action strings, not exact induced probability ratios from Eq. 53 when the projection is many-to-one. The ALFWorld trace in Appendix F.12 gives a concrete example of this gap. This limitation affects the likelihood-based baselines, not ADWM, which does not use importance ratios and conditions its transition model on the executed action a_t.

G.2 Estimator formulas

For a logged trajectory $\tau = (s_0, a_0, r_0, \ldots, s_T, a_T, r_T)$, let ρ_t denote the token-likelihood proxy ratio defined in Appendix G.1, with cumulative ratio $\rho_{0:t} = \prod_{k=0}^{t} \rho_k$ and discount $\gamma = 0.99$:

$$\hat{J}^{\text{DM}} = \frac{1}{N}\sum_\tau \sum_t \gamma^t\, R_\theta(\hat{s}_t) \qquad \text{(direct method, no } \rho\text{)} \tag{54}$$

$$\hat{J}^{\text{IS}} = \frac{1}{N}\sum_\tau \rho_{0:T} \sum_t \gamma^t r_t, \qquad \hat{J}^{\text{WIS}} = \frac{\sum_\tau \rho_{0:T} \sum_t \gamma^t r_t}{\sum_\tau \rho_{0:T}} \tag{55}$$

$$\hat{J}^{\text{FQE}} = \frac{1}{N}\sum_\tau V_{\pi_e^{\varepsilon}}(s_0) \qquad \text{(no per-trajectory } \rho\text{)} \tag{56}$$

$$\hat{J}^{\text{DR}} = \frac{1}{N}\sum_\tau \sum_t \gamma^t [\rho_{0:t}\, r_t + \rho_{0:t-1} V(s_t) - \rho_{0:t}\, Q(s_t, a_t)] \tag{57}$$

with the convention $\rho_{-1} = 1/N$ for the COBS DR_v2 step-form. The DR network values V and Q are taken from the FQE Q-network. The baselines use $\gamma = 0.99$ following COBS, whereas ADWM and the ground truth use the undiscounted episode score; since all methods are compared by Spearman ρ across ε levels, only the ranking of each estimate matters.

G.3 Per-cell importance-ratio variance

Empirically observed cumulative log-ratio magnitudes at ε=0.5 span tens of nats per trajectory (HotpotQA up to ± 24, ScienceWorld ± 18, ALFWorld ± 35, WebShop ± 28), corresponding to

per-trajectory ratios that span $\sim$ 16–30 orders of magnitude. This is the regime in which IS and DR estimates fluctuate by 10^{20} or more even in double-precision arithmetic; our raw baseline output JSONs (released alongside the code) record DR values up to $|J^{\mathrm{DR}}| \approx 6 \times 10^{20}$ on HotpotQA-PRM, exemplifying this divergence. The corresponding rank-correlations become driven entirely by floating-point underflow ties rather than the underlying signal.

G.4 FQE Q-network and per-ε correction

The FQE Q-network is a 2-hidden-layer MLP $Q_\phi : \mathbb{R}^{d_z+d_a} \to \mathbb{R}$ ($d_z{+}d_a \to 128 \to 128 \to 1$, SiLU activations). A target network $Q_{\phi'}$ is updated by Polyak averaging ($\phi' \leftarrow 0.99\,\phi' + 0.01\,\phi$). optimization: Adam, lr 3×10^{-4}, batch size 64, 20 epochs, discount $\gamma = 0.99$. Each training minibatch consists of $(s_t, a_t, r_t, s_{t+1}, \text{done}_t)$ tuples drawn from $\mathcal{D}_b$, and the Bellman target is $y = r_t + \gamma\,(1 - \text{done}_t)\,V_{\pi_e^\varepsilon}(s_{t+1})$ with $V_{\pi_e^\varepsilon}(s) = (1{-}\varepsilon)\max_a Q_{\phi'}(s,a) + \varepsilon\,\bar{Q}_{\phi'}(s)$ where $\bar{Q}_{\phi'}(s)$ is the action-mean across the admissible set. The $\max_a$ approximation is used because the LLM does not expose a full per-action probability vector. It evaluates the greedy policy with respect to $Q_{\phi'}$ rather than the action chosen by π_e, so FQE depends on π_e^ε only through ε and the per-ε data filtering below; this is a limitation of FQE in the LLM-agent setting rather than of ADWM.

Per-ε correction. Under the COBS protocol, mixing is applied at evaluation time as $V_\varepsilon(s) = (1{-}\varepsilon)\max_a Q_{\phi'}(s,a){+}\varepsilon\,\bar{Q}_{\phi'}(s)$, producing a value linear in ε and trivially monotonically decreasing. This artefact gives FQE a spurious $\rho{>}0$ on the ε-axis even when its underlying state-value estimate is incorrect, and inflates the COBS-protocol headline number to a rank that has nothing to do with the policy being scored. We remove this artefact by training a separate Q-network for each ε level using π_b trajectories filtered against the ε-mixed behavior likelihood, then evaluating each on its own $V_{\pi_e^\varepsilon}(s_0)$ at the trajectory start. This is the *per-ε correction* reported in the main text and is the only modification to FQE relative to COBS. Without it, FQE reports a near-uniform $\rho{=}{+}0.9$ on every cell; with it, the true rank correlation collapses or inverts as reported in Table 2.

H Additional Results

H.1 Per-seed Spearman ρ

Table 5 reports the full 5-seed breakdown that underlies the avg-$\hat{J}$ row of Table 2. Per-seed Spearman is computed by independently running each ε cell with seed $s_i \in \{0,1,2,3,4\}$, which propagates randomness through both imagined-rollout sampling and the ε-greedy admissible-action choice. The two reward-dense cells (HotpotQA-DPO, ScienceWorld-ETO) are highly stable across seeds ($\sigma \approx 0.10$); the ALFWorld and WebShop cells show larger per-seed variance because high-ε GT saturates near zero, making the 5-point Spearman ill-conditioned at the seed level. The avg-$\hat{J}$ aggregator used in Table 2 mitigates this by averaging $\hat{J}$ across seeds before computing ρ.

Table 5: Per-seed Spearman ρ between ADWM's $\hat{J}$ and the ground-truth ε-curve (5 ε levels, 5 seeds). The last column reproduces the avg-$\hat{J}$ row from Table 2.

Cell	ρ_{s_0}	ρ_{s_1}	ρ_{s_2}	ρ_{s_3}	ρ_{s_4}	mean$\pm\sigma$	avg-$\hat{J}$ ρ
HotpotQA-DPO	$+1.00$	$+0.80$	$+0.70$	$+0.90$	$+0.90$	$+0.86 \pm 0.10$	$+0.90$
ScienceWorld-ETO	$+0.87$	$+0.87$	$+0.97$	$+0.67$	$+0.82$	$+0.84 \pm 0.10$	$+0.82$
ALFWorld-iter1	$+0.82$	$+0.10$	$+0.67$	-0.10	$+0.21$	$+0.34 \pm 0.35$	$+0.67$
ALFWorld-iter3	-0.50	$+0.90$	$+1.00$	-0.30	$+0.80$	$+0.38 \pm 0.64$	$+0.80$
WebShop-iter1	$+0.60$	-0.10	$+0.90$	$+0.30$	$+1.00$	$+0.54 \pm 0.40$	$+0.90$

H.2 Underlying $\hat{J}$ and GT curves

Table 6 provides the seed-averaged $\hat{J}$ values and ground-truth success curves underlying every row of Table 2, so that any of the reported correlations can be reproduced exactly.

Table 6: Seed-averaged $\hat{J}$ and ground-truth value at each ε for every cell.

Cell	ε=0	0.25	0.5	0.75	1.0
HP-DPO GT	0.165	0.151	0.082	0.095	0.065
HP-DPO $\hat{J}$	0.508	0.496	0.454	0.418	0.364
SW-ETO GT	0.141	0.078	0.047	0.047	0.016
SW-ETO $\hat{J}$	0.589	0.567	0.522	0.474	0.478
ALF-iter1 GT	0.153	0.097	0.056	0.000	0.000
ALF-iter1 $\hat{J}$	0.554	0.547	0.537	0.538	0.541
ALF-iter3 GT	0.250	0.139	0.056	0.028	0.000
ALF-iter3 $\hat{J}$	0.792	0.789	0.789	0.767	0.778
WS-iter1 GT	0.073	0.085	0.073	0.044	0.032
WS-iter1 $\hat{J}$	0.484	0.481	0.466	0.459	0.458

H.3 Training loss and OPE accuracy

We also compared held-out denoising, reward, and termination losses with the downstream OPE error across world-model checkpoints. We found no consistent correlation between these losses and OPE accuracy. Fig. 3 therefore reflects optimization stability, while OPE accuracy is evaluated directly against ground-truth returns in Table 2 and Appendix H.

I Additional Ablations

I.1 Five-seed extension of Fig. 4

Figure 4 reports single-seed ρ values to keep the component story compact. For each variant we also re-ran the full 5-seed protocol on the same three environments and recovered the same qualitative ordering: removing local CFG drops WebShop ρ to $+0.12 \pm 0.08$ across seeds, removing continuation guidance drops it to $+0.27 \pm 0.10$, and removing the ψ adapter drops HotpotQA to $+0.42 \pm 0.07$. No 5-seed run flips the sign of the corresponding single-seed value, and the relative magnitude of each component's contribution is preserved.

I.2 Latent capacity ablation

Beyond the inference-time guidance and adapter terms ablated in Fig. 4, we probe the latent capacity on WebShop, the most data-sensitive of our cells. Halving the latent dimension from d_z=64 to d_z=32 produces a complete inversion ($\rho = -1.00$), suggesting that the latent is at a representational floor below which contrastive features cannot be preserved. Adequate latent capacity must therefore be in place before any choice of guidance can recover positive ranking. We did not observe similar inversions on HotpotQA or ScienceWorld at the same cut, suggesting that the WebShop result reflects the combination of long-horizon search and continuous partial reward rather than a generic property.

I.3 ψ-adapter loss design

Table 7 shows that the loss design we adopt (InfoNCE + $0.1\times$ MSE) is essential. Pure MSE collapses to a near-constant adapter on every environment because per-token-mean target embeddings have low variance, providing the optimiser with no signal to differentiate latents; pure InfoNCE produces a discriminative adapter whose outputs drift out-of-distribution for the LLM and break its downstream input expectations. The combined loss outperforms either alone on all three environments.

Table 7: ψ-adapter loss design ablation, single-seed ρ.

Loss	HotpotQA	ScienceWorld	WebShop
InfoNCE + $0.1\times$ MSE (ours)	$+0.90$	$+1.00$	$+0.90$
Pure MSE	$+0.40$	$+0.40$	$+0.70$
Pure InfoNCE	$+0.60$	$+0.80$	$+0.50$

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 the three contributions (Theorem 1, guided diffusion realization, empirical evaluation), and the experimental results in Section 5 directly support these claims.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

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

   Answer: [Yes]

   Justification: Section 6 and Appendix A discuss the limitations, including policy diversity, offline coverage, ranking versus calibration, cross-step error propagation, GT evaluation quality, and ψ-adapter retraining when switching evaluation LLM families.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

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 theoretical results (Proposition 1, Theorem 1) are stated with full assumptions and proved in Appendices C and D respectively; the guided score derivation is in Appendix E.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

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: Full implementation details including architecture, hyperparameters, training procedure, and inference protocol are provided in Appendices F–G.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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 and raw baseline output JSONs are publicly released; all benchmarks used (HotpotQA, ScienceWorld, ALFWorld, WebShop) are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

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: Section F provides complete training and evaluation details including hardware, optimizer settings, data splits, behavior pool composition, and GT episode counts.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

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: Table 2 reports 95% bootstrap confidence intervals and p-values for Spearman ρ; Table 5 reports per-seed mean $\pm\,\sigma$ across 5 random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

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: Section F.2 specifies hardware (2× RTX 8000, 48 GB), per-job wall-clock times, and total budget (∼300 GPU-hours) with a breakdown by component.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

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: This work involves only publicly released LLM checkpoints and existing benchmarks; no human subjects or sensitive data are involved.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The framework reduces the need for live agent deployment during evaluation, lowering both API cost and the risk of irreversible side effects; no foreseeable negative societal impacts are identified beyond standard ML misuse concerns.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

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: The paper does not release new pretrained models or scraped datasets that pose misuse risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

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: All benchmarks and LLM checkpoints used are cited with their original papers; publicly released checkpoints from prior work are listed in Table 1 with citations.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: The world model code and behavior trajectory pools are publicly released with usage instructions.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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: The paper does not involve crowdsourcing or human subject experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

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: No human subjects are involved.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

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: [Yes]

    Justification: LLMs (π_b and π_e) are core components of the evaluation framework; their roles as behavior and evaluation policies are described throughout Sections 3–5.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.