ROBUSTIFYING DECISION TRANSFORMERS AT TEST TIME WITH DIRECT $\mathcal{L}_1$ COMPENSATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Policies trained offline, such as the Decision Transformer (DT), degrade when the deployment transition dynamics differ from those seen during training. $\mathcal{L}_1$ adaptive control offers a test-time remedy, but its cancellation step is built for control-affine dynamics, so applying it to a learned model has meant affinizing that model and tuning when to refresh the affinization. We propose direct $\mathcal{L}_1$ compensation, an $\mathcal{L}_1$-inspired compensator that uses the learned next-state model directly with no affinization or tolerance needed. Analytically, direct cancellation's undamped first step is a regularized affine inversion, and when the learned model is control-affine and no regularizer is used, the direct cancellation coincides with the affine inversion. On MuJoCo locomotion under matched disturbances, direct $\mathcal{L}_1$ overall achieves better compensation than affine $\mathcal{L}_1$ for DT policies trained across diverse environments, and datasets. Ablation studies locate the primary source of the gain in the removal of the stale affinization action, which is made possible by direct $\mathcal{L}_1$'s non-interrupted state predictor and filter updates. The approach is agnostic to the base policy, so it applies beyond transformer-based policies.

1 INTRODUCTION

Reinforcement learning (RL) underpins many advances in sequential decision-making, spanning robotics (Levine et al., 2016), game-playing (Silver et al., 2017), autonomous driving (Kiran et al., 2021), and large language model post-training (Schulman et al., 2017; Shao et al., 2024; Ouyang et al., 2022; Guo et al., 2025). Within offline RL, sequence modeling has become a dominant lens. The Decision Transformer (DT) (Chen et al., 2021) reframes RL as conditional sequence prediction: given states, actions, and returns-to-go (RTGs), a causal transformer autoregressively predicts actions targeting a desired performance level, sidestepping value estimation and policy gradients in favor of the architectural paradigms that drove progress in language modeling (Vaswani et al., 2017). Follow-up work extends the idea to trajectory-level planning (Janner et al., 2021), multi-task generalization (Lee et al., 2022), online fine-tuning (Zheng et al., 2022), and trajectory stitching (Wu et al., 2023).

However, policies trained offline inevitably face distribution shifts at deployment. Actuator degradation, unmodeled contact, payload changes, and other perturbations of the transition dynamics produce state-action trajectories that lie outside the support of the training data, causing learned policies to degrade or fail unpredictably. Existing robustification of DT-based methods operates entirely at the training level: RDT (Xu et al., 2025) mitigates data corruption through embedding dropout, loss reweighting, and iterative data correction, while ARDT (Tang et al., 2024) relabels trajectory returns with in-sample minimax estimates via expectile regression to condition on worst-case returns. Both improve robustness within their respective scopes, but neither provides an online corrective mechanism that senses and counteracts a discrepancy that only appears at deployment.

Adaptive control offers exactly such a mechanism. The $\mathcal{L}_1$ adaptive control ($\mathcal{L}_1$AC) architecture (Hovakimyan & Cao, 2010) estimates a lumped uncertainty from a prediction error and cancels it through a bandwidth-limited additive input, and it has been combined with learned dynamics for model-based RL (Cheng et al., 2022; Sung et al., 2024). Its estimation step asks nothing of the model's form, since the predictor only needs the model's prediction at the commanded action. Its cancellation step, however, is built for *control-affine* dynamics: it splits the estimate into the part an input matrix can reach and the part it cannot. A learned next-state model has no input matrix, so applying $\mathcal{L}_1$AC to it has meant affinizing the model in the action about a reference action, inverting

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

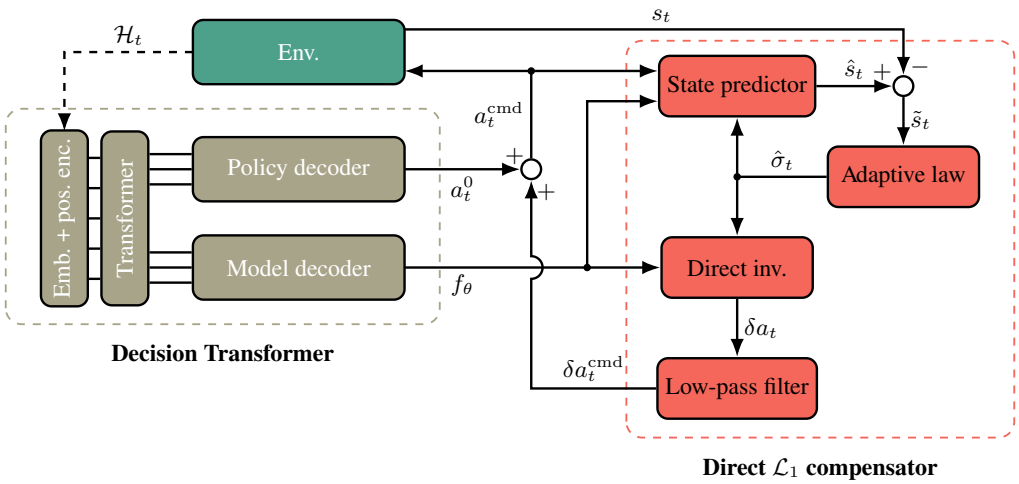

Figure 1: **Direct $\mathcal{L}_1$ test-time compensation of a DT.** The DT's policy decoder proposes the nominal action a_t^0, and its model decoder is read as a next-state model f_θ that is nonlinear in the action. The compensator wraps the estimation components of $\mathcal{L}_1$AC around this model. The state predictor advances $\hat{s}_t$ from the commanded action and the current estimate, the prediction error $\tilde{s}_t = \hat{s}_t - s_t$ drives the adaptive law, and its output $\hat{\sigma}_t$ feeds both the predictor and the inversion. The direct inversion solves a regularized optimization problem for the correction δa_t that cancels the one-step residual through the model's full action effect. The low-pass filter yields $\delta a_t^{\mathrm{cmd}}$, which is added to a_t^0 to form the command a_t^{cmd}. The command is applied to the environment and, causally, to the inference process at the next step.

the resulting Jacobian, and refreshing the affinization, state predictor, and filter whenever its error exceeds a tolerance (Sung et al., 2024). Sung et al. (2024) already discuss the tolerance tradeoff in their limitations: a small ϵ_a suppresses the compensation, a large one admits more affinization and predictor error.

Contributions. We propose a test-time compensator for a DT, inspired by the $\mathcal{L}_1$ adaptive control framework, that uses the learned nonlinear-in-action next-state model directly. We remove the surrogate and its threshold which largely restrict $\mathcal{L}_1$'s full capability on exploiting learned dynamics model, so surrogate accuracy is no longer traded against restarting the estimator. We prove that direct cancellation's undamped first step is a regularized affine inversion, and when the learned model is control-affine and no regularizer is used, the correction we solve for is exactly the affine cancellation. On diverse environments, datasets, and action-channel disturbances, we demonstrate that direct $\mathcal{L}_1$ proposes better compensation than affine $\mathcal{L}_1$ for DT policies. We further conduct ablation studies, indicating that the removal of the stale affinization action is the primary source of the performance gains of direct $\mathcal{L}_1$.

2 Preliminaries

2.1 Decision Transformer

DT (Chen et al., 2021) reframes RL as a conditional sequence modeling task, using a causal transformer (Vaswani et al., 2017) to predict actions from a sequence of past states, actions, and returns-to-go (RTGs). A trajectory is represented as $\tau = (R_0, s_0, a_0, \ldots, R_T, s_T, a_T)$, where s_t is the state, a_t the action, $R_t = \sum_{t'=t}^{T} r_{t'}$ the undiscounted RTG at timestep t, and T the episode length. Inputs are projected into a shared latent space via modality-specific linear embeddings, summed with learned timestep embeddings, and processed by a causal transformer that autoregressively predicts actions. At inference a target return R_0 is specified and the RTG is decremented by observed rewards, $R_{t+1} = R_t - r_t$. We leave this update untouched throughout.

The architecture accommodates auxiliary prediction heads without degrading policy performance (Chen et al., 2021). We train a *three-head* model $\mathcal{M}_\theta$ that reads the same causal backbone and emits the nominal action $\pi_\theta(\mathcal{H}_t)$, a next-state prediction $\hat{f}_\theta(\mathcal{H}_t, a_t)$, and a next-RTG prediction $\hat{R}_\theta(\mathcal{H}_t, a_t)$, where $\mathcal{H}_t := (\tau_{t-K+1:t-1}, R_t, s_t)$ is the context over the last K timesteps, up to and including the current state. Tokens are ordered (R, s, a) within each step, so the policy head reads the backbone at the state token and never sees a_t, while the state and return heads read it at the action token, one position later; the action is therefore an explicit argument of those two heads. The model is trained with the three-term loss

$$\mathcal{L} := \mathbb{E}_t \left[\|\pi_\theta(\mathcal{H}_t) - a_t\|_2^2 + \mu \|\hat{f}_\theta(\mathcal{H}_t, a_t) - s_{t+1}\|_2^2 + \nu \|\hat{R}_\theta(\mathcal{H}_t, a_t) - R_{t+1}\|_2^2 \right], \tag{1}$$

where μ is the state weight and ν is the RTG weight.

All three heads are trained once, offline, and are not updated thereafter. In our work, both policy and state heads are required. The policy head provides the nominal action, $a_t^0 := \pi_\theta(\mathcal{H}_t)$, and the state head evaluated at an arbitrary action, $\hat{f}_\theta(\mathcal{H}_t, a)$, is the context-conditioned next-state map that the compensator uses as a learned one-step dynamics model. The RTG head is not queried at test time and thus is optional.

2.2 $\mathcal{L}_1$ ADAPTIVE CONTROL

$\mathcal{L}_1$AC provides transient and steady-state guarantees under the structural assumptions, uncertainty bounds, and filter conditions of the applicable results (Hovakimyan & Cao, 2010). It has been validated on robotic (Cheng et al., 2022; Pravitra et al., 2020; Wu et al., 2022b) and aerial platforms (Gregory et al., 2009; Ackerman et al., 2017) and combined with data-driven models (Sung et al., 2024; Gahlawat et al., 2025). Its formulation assumes control-affine dynamics of the form

$$\dot{s}(t) = g(s(t)) + h(s(t))\, a(t) + d(t, s(t), a(t)), \tag{2}$$

with state $s \in \mathbb{R}^n$, action $a \in \mathbb{R}^m$, known nominal terms g, h, and unknown residual d. The architecture has three parts: a *state predictor* $\dot{\hat{s}} = g(s) + h(s)a + \hat{\sigma} + A_s\tilde{s}$, with predictor state $\hat{s}$, prediction error $\tilde{s} = \hat{s} - s$, and A_s Hurwitz (every eigenvalue has negative real part), so that the prediction error decays exponentially; a piecewise-constant *adaptive law* $\hat{\sigma}(iT_s) = -\Phi^{-1}(T_s)\mu(iT_s)$ with $\Phi(T_s) = A_s^{-1}(e^{A_s T_s} - I)$ and $\mu(iT_s) = e^{A_s T_s}\tilde{s}(iT_s)$, converting that error into an uncertainty estimate; and a *low-pass filter* $C(s)$ producing the additive input $\mathfrak{L}[\tilde{a}] = -C(s)\mathfrak{L}[\hat{\sigma}_m]$, which restricts the adaptive signal to low frequencies and decouples the fast estimation loop from the control channel.

The step that matters here is the split $\hat{\sigma} = h(s)\hat{\sigma}_m + h^\perp(s)\hat{\sigma}_{um}$, where $h^\perp$ spans the orthogonal complement of the column space of h. Only the *matched* part $h(s)\hat{\sigma}_m$ lies in the range of the input matrix and can be cancelled; the *unmatched* part $h^\perp(s)\hat{\sigma}_{um}$ cannot be rejected by any input. This is the classical matching condition, and it presupposes that the input enters *linearly*, through $h(s)$.

A learned next-state model $\hat{f}_\theta(s, a)$ has no such input matrix. To reuse the split, Sung et al. (2024) affinize the learned model about a reference action $\bar{a}$,

$$\hat{f}_\theta(s, a) \approx \hat{g}_\theta(s, \bar{a}) + \hat{h}_\theta(s, \bar{a})\, a, \quad \hat{h}_\theta(s, \bar{a}) := \tfrac{\partial \hat{f}_\theta}{\partial a}(s, \bar{a}), \quad \hat{g}_\theta(s, \bar{a}) := \hat{f}_\theta(s, \bar{a}) - \hat{h}_\theta(s, \bar{a})\, \bar{a}, \tag{3}$$

which is the first-order Taylor expansion of $\hat{f}_\theta$ in a about $\bar{a}$: the Jacobian $\hat{h}_\theta$ plays the role of the input matrix, and the offset $\hat{g}_\theta$ absorbs everything that does not depend on a, so the surrogate matches $\hat{f}_\theta$ in value and in first derivative at $a = \bar{a}$. In our DT baseline, the coefficients are recomputed in the current context at every step; the reference action is moved when the induced error exceeds ϵ_a. The surrogate is then treated as equation 2. This is the construction we take as our affine baseline, and the construction our method removes.

3 METHOD

The compensator wraps a trained DT and performs only test-time compensation. It reads the state head $\hat{f}_\theta$ as a next-state model and uses it as a general nonlinear function of the action. At every timestep it (i) estimates the transition residual causally, from the previously commanded action, with an $\mathcal{L}_1$ state predictor and piecewise-constant adaptive law (Section 3.1); (ii) converts that estimate

into an action correction by inverting the *full nonlinear action effect* of that model (Section 3.2); and (iii) passes the correction through a low-pass filter before the command is issued (Section 3.3); Figure 1 shows the loop.

3.1 CAUSAL RESIDUAL ESTIMATION

Let a_{t-1}^{cmd} be the action actually commanded at the previous step. The one-step prediction error, the gap between the observed s_t and the model's prediction $\hat{f}_\theta(\mathcal{H}_{t-1}, a_{t-1}^{\mathrm{cmd}})$, is causal, requires no additional model, and lumps together the deployment-time transition discrepancy and the error of the learned model itself. We turn it into an estimate with the two components $\mathcal{L}_1$AC uses for the purpose, a state predictor and a piecewise-constant adaptive law (Section 2.2).

The nonlinear learned dynamics enter the state predictor directly instead of through a control-affine surrogate. With predictor state $\hat{s}$, the finite-difference drift $\hat{\varphi}_t(a) := \big(\hat{f}_\theta(\mathcal{H}_t, a) - s_t\big)/T_s$, the uncertainty estimate $\hat{\sigma}$ supplied by the adaptive law introduced next (equation 5), a Hurwitz $A_s = \alpha_s I$, $\alpha_s < 0$, chosen by the user, and prediction error $\tilde{s} := \hat{s} - s$, the *state predictor* is

$$\dot{\hat{s}}(t) = \hat{\varphi}_t\big(a_t^{\mathrm{cmd}}\big) + \hat{\sigma}(t) + A_s\,\tilde{s}(t), \qquad \hat{s}(0) = s_0. \tag{4}$$

It is driven by the commanded action rather than the nominal one, so it tracks the deployed loop.

The state predictor induces the same piecewise-constant adaptive law as in Section 2.2 (See derivation in Section B.1),

$$\hat{\sigma}_t = -\Phi^{-1}(T_s)\, e^{A_s T_s}\,\tilde{s}_t, \qquad \Phi(T_s) = A_s^{-1}\big(e^{A_s T_s} - I\big), \qquad \hat{\sigma}_0 = 0. \tag{5}$$

We work throughout with the one-step displacement estimate $\hat{d}_t := T_s\,\hat{\sigma}_t$, which lives in the units of a one-step state displacement.

Algorithm 1 advances equation 4 by one forward-Euler step of length T_s, which telescopes to $\hat{s}_{t+1} = \hat{f}_\theta(\mathcal{H}_t, a_t^{\mathrm{cmd}}) + (I + T_s A_s)\tilde{s}_t + \hat{d}_t$. Under exact continuous-time integration, the adaptive law cancels the contribution of the preceding prediction error while retaining the interval's disturbance integral. The Euler update additionally retains $M\tilde{s}_t$, with leading coefficient $\alpha_s T_s/2$ (Section B.1).

3.2 DIRECT ACTION-EFFECT INVERSION

Given the residual estimate $\hat{d}_t$, we convert it into an action correction: the perturbation δa whose addition to the nominal action best cancels $\hat{d}_t$, in the role that the adaptive input $\tilde{a}$ plays in Section 2.2. The predicted effect of that perturbation on the next state is

$$\Delta\hat{f}_t(\delta a) := \hat{f}_\theta(\mathcal{H}_t, a_t^0 + \delta a) - \hat{f}_\theta(\mathcal{H}_t, a_t^0), \tag{6}$$

evaluated by two forward passes and exact for finite δa, and the perturbation solves

$$\delta a_t^\star = \operatorname*{arg\,min}_{\delta a \in \mathbb{R}^m} J_t(\delta a), \qquad J_t(\delta a) := \tfrac{1}{2}\big\|\Delta\hat{f}_t(\delta a) + \hat{d}_t\big\|_Q^2 + \tfrac{\lambda}{2}\|\delta a\|_R^2, \tag{7}$$

with $Q, R \succ 0$ diagonal. The first term is the compensation objective, written with the *exact* finite action effect of equation 6; the second penalizes a large correction.

We numerically solve equation 7 with Levenberg–Marquardt (Nocedal & Wright, 2006). Let $r_t(\delta a) := \Delta\hat{f}_t(\delta a) + \hat{d}_t$ and $J_r := \partial r_t/\partial\delta a$, taken by automatic differentiation. Starting from $\delta a^{(0)} = 0$, each iteration forms the damped normal equations

$$\big(J_r^\top Q J_r + \lambda R + \mu_k I\big)\, p^{(k)} = -\nabla J_t(\delta a^{(k)}) = -\big(J_r^\top Q\, r_t(\delta a^{(k)}) + \lambda R\, \delta a^{(k)}\big). \tag{8}$$

The iteration backtracks over $\gamma \in \{1, \frac{1}{2}, \frac{1}{4}, \ldots\}$, accepting $\delta a^{(k+1)} = \delta a^{(k)} + \gamma p^{(k)}$ only if it satisfies an Armijo decrease of the objective J_t. On acceptance μ_k is decreased, otherwise increased and the step retried. If the retry budget is exhausted, the solve returns the last accepted iterate. If no trial has been accepted, the initial zero iterate is retained. The solve is capped at K_{it} accepted steps in total, and we write δa_t for what it returns, which is $\delta a_t^\star$ only when the iteration reaches the minimizer.

Algorithm 1 Direct $\mathcal{L}_1$ test-time compensation of a DT

Input: trained DT $\mathcal{M}_\theta$, environment `env`, context length K, horizon T, target return R_0; sampling interval T_s, observer pole α_s; filter cutoff ω_c; weights Q, R, λ; actuator bounds $(a_{\text{lo}}, a_{\text{hi}})$; iteration cap K_{it}
$s_0 \leftarrow$ `env.reset()`; $\tau \leftarrow ()$; $\hat{s}_0 \leftarrow s_0$; $\delta a_{-1}^{\text{cmd}} \leftarrow 0$; $a_{-1}^{\text{cmd}} \leftarrow 0$
for $t = 0, 1, \ldots, T-1$ **do**
  $\mathcal{H}_t \leftarrow (\tau_{t-K+1:t-1}, R_t, s_t)$ ▷ context of Section 2.1
  $a_t^0 \leftarrow \pi_\theta(\mathcal{H}_t)$ ▷ policy head
  $\tilde{s}_t \leftarrow \hat{s}_t - s_t$; $\hat{\sigma}_t \leftarrow$ adaptive law (equation 5); $\hat{d}_t \leftarrow T_s \hat{\sigma}_t$
  $\delta a_t \leftarrow$ LM on the objective in equation 7, at most K_{it} accepted steps
  Apply low-pass filter to δa_t to get δa_t^{cmd} (equation 9)
  $\delta a_t^{\text{cmd}} \leftarrow \text{clip}(\delta a_t^{\text{cmd}}, a_{\text{lo}} - a_t^0, a_{\text{hi}} - a_t^0)$; $a_t^{\text{cmd}} \leftarrow a_t^0 + \delta a_t^{\text{cmd}}$
  $s_{t+1}, r_t, \texttt{done} \leftarrow \texttt{env.step}(a_t^{\text{cmd}})$
  $R_{t+1} \leftarrow R_t - r_t$ ▷ DT return-to-go update
  $\hat{s}_{t+1} \leftarrow \hat{f}_\theta(\mathcal{H}_t, a_t^{\text{cmd}}) + (I + T_s A_s)\tilde{s}_t + T_s \hat{\sigma}_t$; extend τ by $(R_t, s_t, a_t^{\text{cmd}})$ ▷ forward Euler
  if `done` **then break**
end for

3.3 Bandwidth limiting

Following the separation of estimation and actuation in classical $\mathcal{L}_1$AC (Hovakimyan & Cao, 2010; Sung et al., 2024), we filter the computed action correction before clipping. The residual estimate contains model error and any measurement noise as well as deployment mismatch, so applying the correction unfiltered can pass rapid fluctuations to the actuators. The low-pass filter attenuates these fluctuations. It is applied to the correction the solver returns, and the clip of Algorithm 1 then gives the commanded correction δa_t^{cmd} that the recursion carries forward:

$$\delta a_t^{\text{cmd}} = \beta\, \delta a_{t-1}^{\text{cmd}} + (1-\beta)\, \delta a_t, \qquad \beta = e^{-\omega_c T_s}, \tag{9}$$

where ω_c is the cutoff frequency, which trades disturbance tracking against noise amplification.

The full evaluation loop is Algorithm 1.

4 Analysis

The propositions relate the ideal Gauss–Newton step and the affine objective's minimizer to affine compensation; proofs are in Section B.2.

Assumption 1 (Nominal-action differentiability)**.** For fixed $\mathcal{H}_t$, the map $a \mapsto \hat{f}_\theta(\mathcal{H}_t, a)$ is differentiable at the nominal action a_t^0.

Proposition 1 (The undamped first step solves the regularized affine model)**.** *Under Assumption 1, let $Q, R \succ 0$ and $\lambda > 0$. Linearizing the residual r_t at $\delta a = 0$ gives $q_t(\delta a) := \frac{1}{2}\|\hat{d}_t + J_{a,t}\, \delta a\|_Q^2 + \frac{\lambda}{2}\|\delta a\|_R^2$, where $J_{a,t} := J_a \hat{f}_\theta(\mathcal{H}_t, a_t^0)$. It is strictly convex, and its unique minimizer $\delta a_t^{\text{gn}} = -(J_{a,t}^\top Q J_{a,t} + \lambda R)^{-1} J_{a,t}^\top Q\, \hat{d}_t$ is the undamped Gauss–Newton update of equation 8 from $\delta a^{(0)} = 0$, with $\mu_0 = 0$ and unit step length.*

Proposition 1 identifies the ideal undamped full step with a regularized affine correction: the Gauss–Newton model replaces the action effect by its Jacobian at the nominal action, and the regularizer makes the inversion a ridge, with $J_{a,t}$ in the role of the input matrix.

Proposition 2 (Exact reduction to the affine inversion)**.** *Suppose the state head is control-affine in the action on $\mathbb{R}^m$, so that the affinization equation 3 is exact rather than approximate: $\hat{f}_\theta(\mathcal{H}_t, a) = \hat{g}_\theta(\mathcal{H}_t) + \hat{h}_\theta(\mathcal{H}_t)\, a$, with $\hat{h}_t := \hat{h}_\theta(\mathcal{H}_t) \in \mathbb{R}^{n \times m}$ of full column rank. Take $Q = I$ and $\lambda = 0$. Then J_t is a strictly convex quadratic, and its unique minimizer is*

$$\delta a_t^\star \;=\; -\big(\hat{h}_t^\top \hat{h}_t\big)^{-1} \hat{h}_t^\top \hat{d}_t \;=\; -\hat{h}_t^\dagger \hat{d}_t \;=\; -\hat{\sigma}_{m,t}. \tag{10}$$

Table 1: **Action-channel attack, $k = 1$.** D4RL-normalized returns under random-walk action noise: mean $\pm$ standard deviation across nine checkpoint–seed pairs, each a ten-episode mean. Bold marks the highest mean per row. Direct has higher mean return than DT in every cell and than affine $\mathcal{L}_1$ in all but hopper `medium`.

Env	Dataset	DT	Affine $\mathcal{L}_1$	Direct $\mathcal{L}_1$ (ours)
halfcheetah	expert	66.1 ± 2.4	67.4 ± 3.4	$\mathbf{72.1 \pm 1.8}$
	medium	29.2 ± 2.5	28.5 ± 3.7	$\mathbf{34.6 \pm 0.7}$
	medium-replay	31.1 ± 2.6	29.5 ± 2.8	$\mathbf{33.9 \pm 2.1}$
hopper	expert	85.9 ± 5.0	81.1 ± 7.2	$\mathbf{88.1 \pm 7.5}$
	medium	43.6 ± 2.9	$\mathbf{46.7 \pm 4.2}$	45.8 ± 4.2
	medium-replay	56.9 ± 8.8	52.3 ± 14.2	$\mathbf{61.5 \pm 11.2}$
walker2d	expert	78.7 ± 7.4	79.1 ± 9.9	$\mathbf{83.1 \pm 6.1}$
	medium	47.0 ± 4.8	50.6 ± 2.7	$\mathbf{53.6 \pm 3.7}$
	medium-replay	36.0 ± 5.9	37.7 ± 4.2	$\mathbf{42.9 \pm 5.2}$

Proposition 2 says that when the state head is control-affine, the unregularized objective has the closed-form affine inversion as its unique minimizer, and that the solver reproduces the baseline's cancellation in a single step when that step is undamped and taken in full.

Remark 1 (Regularized inversion in the control-affine case)**.** Under the hypotheses of Proposition 2 but with $\lambda > 0$, equation 7's unique minimizer is exactly the expression of Proposition 1 at $J_{a,t} = \hat{h}_t$. Taking $Q = R = I$ and writing $\hat{h}_t = U\Sigma V^\top$ with singular values $\varsigma_1, \ldots, \varsigma_m > 0$,

$$\delta a_t^\star \;=\; -V \operatorname{diag}\!\Big(\tfrac{\varsigma_i^2}{\varsigma_i^2+\lambda}\Big)\, V^\top \hat{\sigma}_{m,t}, \tag{11}$$

so the regularizer returns the affine $\mathcal{L}_1$ correction of equation 10 with each singular direction shrunk by a factor in $(0, 1)$, most strongly where ς_i is smallest: exactly the directions in which inverting an ill-conditioned Jacobian amplifies. Letting $\lambda \to 0$ recovers equation 10, and $\lambda \to \infty$ recovers the zero correction.

5 Experiments

We evaluate on HalfCheetah, Hopper, and Walker2d through the Gymnasium interface (Towers et al., 2024), with DTs trained on the D4RL (Fu et al., 2020) `expert`, `medium`, and `medium-replay` datasets. The disturbance is a *matched* transition uncertainty on the executed input, $a_t^{\mathrm{exec}} = a_t^{\mathrm{cmd}} + \varepsilon_t^a$, which is action-compensable by construction. Section C.1 gives the full experimental setup.

Baselines. **DT** (Chen et al., 2021) is the DT policy alone, with no correction. **Affine $\mathcal{L}_1$** applies the correction of Sung et al. (2024) to the DT state head: local affinization as in equation 3 with tolerance ϵ_a, a matched/unmatched split along $\mathrm{range}(\hat{h}_\theta)$, and closed-form inversion of the matched part (Algorithm 2 in Section A). **Direct $\mathcal{L}_1$** is the full method (Algorithm 1), at the solver cap $K_{\mathrm{it}} = 1$ unless stated otherwise.

5.1 Does direct $\mathcal{L}_1$ achieve better compensation?

Table 1 reports the three arms at the calibrated magnitude $k = 1$ under random-walk action noise. Direct $\mathcal{L}_1$ is ahead of DT in all nine cells and of affine $\mathcal{L}_1$ in eight, the exception being hopper `medium`, where affine $\mathcal{L}_1$ is narrowly ahead. Affine $\mathcal{L}_1$ is flat against DT, ahead in about half the cells and behind in the rest.

We also sweep the noise scale k from zero to twice the calibrated magnitude on the three expert cells (Figure 2). At $k = 0$ the arms coincide: the correction costs nothing when there is nothing to correct. The margin of direct $\mathcal{L}_1$ over DT opens at the calibrated magnitude and peaks at $k = 1.5$ on every plant, at 6 to 8 points; at the largest scale it holds on halfcheetah, the plant that cannot fall, and narrows on hopper and walker2d. Affine $\mathcal{L}_1$ tracks DT across the sweep except on hopper at the calibrated magnitude, where it falls short of DT.

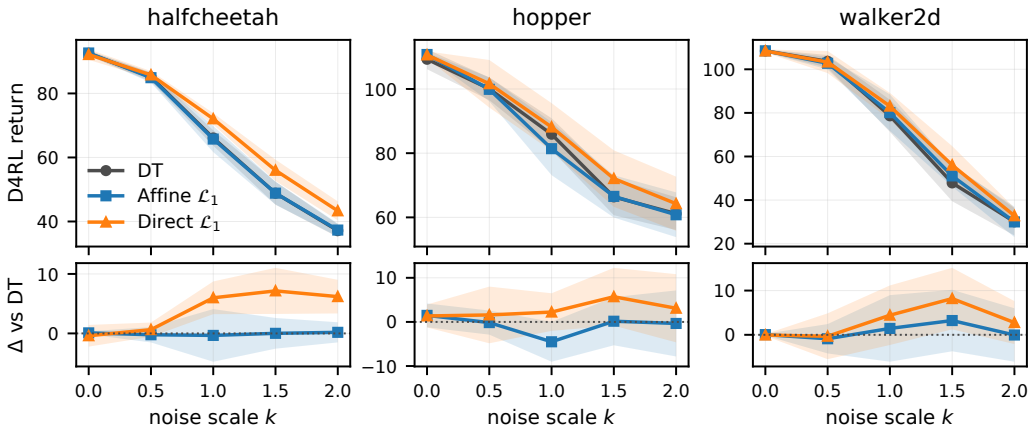

Figure 2: **Magnitude sweep, expert cells.** D4RL-normalized return against the random-walk action-noise scale k (top), and the same comparison as a paired difference against DT (bottom).

Table 2: **Solver-budget ablation.** Time is controller wall time per control step in ms. K_{it} caps the accepted LM steps; *parallel* and *orthogonal*, the correction along and across the disturbance relative to its size, are defined in Section 5.2. One step costs what affine $\mathcal{L}_1$ does; more steps buy nothing on halfcheetah or hopper and 1.8 points on walker2d, and neither diagnostic moves with the budget.

Env		Affine $\mathcal{L}_1$	Direct $\mathcal{L}_1$, $K_{\text{it}} = 1, 2, 3$		
			1	2	3
halfcheetah	return	66.8	72.1	70.7	71.1
	time (ms)	4.47	4.30	6.74	9.21
	parallel	0.02	0.25	0.23	0.24
	orthogonal	0.12	0.27	0.24	0.26
hopper	return	81.1	88.1	85.3	86.8
	time (ms)	4.00	4.11	6.57	8.58
	parallel	0.01	0.14	0.13	0.14
	orthogonal	0.08	0.13	0.13	0.13
walker2d	return	79.1	83.1	84.9	82.8
	time (ms)	4.48	4.89	7.62	10.66
	parallel	0.01	0.10	0.10	0.10
	orthogonal	0.20	0.11	0.12	0.12

5.2 Is direct $\mathcal{L}_1$ necessarily more expensive at test time?

Table 2 reports the wall time per control step. Sweeping $K_{\text{it}} \in \{1, 2, 3\}$ on the three expert cells, one step is as good as three on halfcheetah and hopper and 1.8 points behind two steps on walker2d, while costing what affine $\mathcal{L}_1$ does and about half of what three steps cost.

Table 2 also reports two diagnostics of the correction itself, computed over each rollout against the injected disturbance ε_t^a. Both pool the rollout's steps rather than averaging per-step ratios, which weights each step by how much it moved the plant. Both measure the correction against the size of the disturbance, so 1 is a correction as large as the disturbance itself. *Parallel* is how much of the disturbance the correction cancels, parallel $:= -\sum_t \langle \delta a_t, \varepsilon_t^a \rangle / \sum_t \|\varepsilon_t^a\|^2$; *orthogonal* is what the correction adds in directions the disturbance does not occupy, orthogonal $:= (\sum_t \|\delta a_t - \pi_t \delta a_t\|^2 / \sum_t \|\varepsilon_t^a\|^2)^{1/2}$ with π_t the projection onto ε_t^a. Across the sweep of K_{it} both move by at most 0.03 in every environment: the solver spends more calls without changing the correction it returns, which is why the return does not move either.

Table 3: **Component ablation.** Random-walk action noise on halfcheetah `expert` at $k = 1$, hopper `medium-replay` at $k = 2$, and walker2d `medium` at $k = 1.5$. The affine arm is given the direct arm's ingredients one at a time: an estimator carried across re-affinizations instead of rebuilt, which leaves the state predictor and the correction filter both running through a refresh so that only the surrogate moves; the regularizer; and both. *Zero* is the fraction of steps on which the applied correction is exactly 0; *parallel* and *orthogonal* are defined in Section 5.2.

		restart		correction vs. disturbance		
Env-dataset	arm	refresh	zero	parallel	orthogonal	return
halfcheetah expert, $k = 1$	affine $\mathcal{L}_1$	0.57	0.57	0.19	0.49	67.4
	+carry	0.50	0.00	0.56	0.84	57.3
	+regularizer	0.58	0.58	0.15	0.37	68.5
	+carry +regularizer	0.58	0.00	0.45	0.54	68.4
	direct $\mathcal{L}_1$	n/a	0.00	0.44	0.36	73.0
hopper medium-replay, $k = 2$	affine $\mathcal{L}_1$	0.03	0.03	0.15	0.36	40.3
	+carry	0.03	0.00	0.24	0.53	41.8
	+regularizer	0.03	0.03	0.12	0.17	42.0
	+carry +regularizer	0.03	0.00	0.22	0.25	45.9
	direct $\mathcal{L}_1$	n/a	0.00	0.22	0.22	50.0
walker2d medium, $k = 1.5$	affine $\mathcal{L}_1$	0.03	0.03	0.12	0.51	38.9
	+carry	0.03	0.00	0.18	0.74	37.1
	+regularizer	0.03	0.03	0.06	0.17	40.2
	+carry +regularizer	0.03	0.00	0.13	0.26	43.4
	direct $\mathcal{L}_1$	n/a	0.00	0.15	0.16	45.2

5.3 What is the mechanism behind the performance differences?

Within Table 3 the arms are matched except for the ingredient each row names, at settings of their own (Table 8). Three distinctions motivate the ablation. (i) *The action-effect model.* The baseline affinizes the state head about a reference action and uses that surrogate in both the state predictor and the inversion. Its coefficients are evaluated in the current context on every step, while the reference action moves when the affinization error exceeds ϵ_a; ours uses the learned model in both places, with the first inversion step differentiated at the current nominal action. Besides, the same two models set the predictor's drift, the baseline integrating the surrogate at the applied command and ours the state head there, so the estimate handed to the inversion differs too. (ii) *The lifetime of the estimator.* In the switching construction of Sung et al. (2024) that we adopt, a reference-action refresh rebuilds the estimator: the predictor is re-anchored to the measured state and the correction filter is cleared, so the correction is zero on that step; ours resets once per episode. (iii) *Whether the correction is regularized.* The baseline's closed-form inversion cancels the entire matched estimate; ours approximately minimizes equation 7, whose regularizer, weighted by λ, penalizes a large correction.

Carry alone has mixed effects on mean return. It removes the reset-induced zero correction at reference refreshes, and both correction diagnostics increase in all three cells. Mean return changes by -10.1 on halfcheetah, where +carry refreshes on about half the steps, $+1.5$ on hopper and -1.8 on walker2d. The ridge alone lowers both correction diagnostics and raises mean return by 1.1, 1.7 and 1.3: a smaller correction cancels less of the disturbance but also sends less of it sideways. The pair raises it by 1.0, 5.6 and 4.5. Adding carry to the regularized arm changes mean return by -0.1, $+3.9$ and $+3.2$, respectively. Thus these differences do not provide a unique additive attribution of the direct arm's advantage.

Most of what separates +both from direct is how stale the linearization action is. Since +both closes (ii) and (iii), what remains is (i) and the solver, which Table 10 splits three ways on matched entries. Moving the linearization point to the current nominal action at every step, which is $\epsilon_a \to 0$ with the estimator carried, recovers $+4.2$, $+2.6$ and $+1.2$ of gaps of 4.6, 4.1 and 1.9, and carries the diagnostics with it: the orthogonal share falls 0.11, 0.03 and 0.10 of totals of 0.18, 0.03 and 0.10, the parallel share staying within 0.04. What the stale action costs is the correction's direction, not its size. Giving the predictor the state head instead adds $+1.2$, $+1.2$ and $+0.7$ and at most 0.04 more, so we read it as secondary; with the reference fresh and $K_{\text{it}} = 1$ the two inversions coincide, so that

step moves the predictor alone, and Table 2 finds further accepted iterations buy nothing; the solver's damping and backtracking move nothing measurable.

6 Related Work

Robust offline RL. Training-time approaches modify the learning procedure (Zhang et al., 2022; Ye et al., 2023; Yang et al., 2024; Wu et al., 2022a). Test-time approaches learn conservative-smoothed policies (Yang et al., 2022), optimize over distributional uncertainty sets (Shi & Chi, 2022; Panaganti et al., 2022), denoise observations with a diffusion predictor (Yang & Xu, 2024), or learn adversarial dynamics models (Rigter et al., 2022); Conservative Q-Learning (Kumar et al., 2020) penalizes out-of-distribution actions without targeting robustness. All either pay their cost at training time or act on the observation rather than the actuation channel; ours leaves the policy untouched and corrects the commanded action from a causal prediction residual.

Control-theoretic augmentation of learned policies. Cheng et al. (2023) and Chee et al. (2023) pair disturbance observers or $\mathcal{L}_1$ compensation with barrier functions and learning-based MPC, both requiring partial prior dynamics knowledge; Cheng et al. (2022) first integrated $\mathcal{L}_1$ adaptive control with RL using a known model, and Sung et al. (2024) removed that requirement by affinizing a learned predictor and inverting the resulting input matrix. Every one routes the uncertainty estimate through a *linear* inversion. We remove that step: the learned model is inverted as the nonlinear map it is, which also dispenses with the affinization tolerance and its switching logic.

Decision Transformer variants. Beyond DT (Chen et al., 2021) and Trajectory Transformer (Janner et al., 2021), placed by Emmons et al. (2021) in a broader RL-via-supervised-learning paradigm, extensions include Elastic DT (Wu et al., 2023), Q-Learning DT (Yamagata et al., 2023), ESPER (Paster et al., 2022), LaMo (Shi et al., 2024), and DeFog (Hu et al., 2023); RDT (Xu et al., 2025) and ARDT (Tang et al., 2024) target robustness through data correction and minimax return relabeling. All act at training time; none senses a deployment-time transition discrepancy online or inverts the auxiliary state head as a dynamics model.

7 Limitations and Conclusion

Limitations. Our method is inherently tied with the base policy and compensates only what the action channel reaches, so underactuated directions remain uncompensated. Besides, the correction is chosen to cancel the next residual without considering the steps after it, and the actuator bounds enter only through the clip, after the solve has returned. That last point is where the formulation points past itself: since the correction already comes from an optimization problem, the actuator bounds can enter it as constraints, and the same objective extends over a horizon, with the model rolled forward and the correction at each step chosen jointly with those it commits the following steps to.

Conclusion. Inspired by $\mathcal{L}_1$ adaptive control, we present direct $\mathcal{L}_1$, a test-time model-based compensator for offline policies instantiated by a Decision Transformer. Dropping the affine surrogate drops the tolerance with it, so surrogate accuracy is no longer traded against restarting the estimator. The analysis relates direct and affine cancellation: the ideal undamped step of our inversion is a ridge-regularized affine correction at the nominal action, and it is exactly the classical matched cancellation when the head is control-affine and the correction unregularized. On diverse environments, datasets, and action-channel disturbances, we demonstrate that direct $\mathcal{L}_1$ proposes overall better compensation than affine $\mathcal{L}_1$ for DT policies. An ablation put most of that gain in the removal of the stale linearization action.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, assisting with translation, cleaning and reformatting dataset, supporting qualitative and thematic data analysis, or interpreting results. Additionally, we used generative AI tools for drafting parts of a research paper. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. LLM-generated text was reviewed and edited for accuracy and clarity. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies the test-time robustness of offline RL policies on simulated MuJoCo locomotion benchmarks. All datasets used are publicly released D4RL benchmarks. We identify no direct ethical risks and are committed to responsible dissemination, supporting the ethical use of robotic data and AI technologies.

REPRODUCIBILITY STATEMENT

The experimental setup and evaluation protocol are described in Section C.1. Source code will be released.

REFERENCES

Kasey A Ackerman, Enric Xargay, Ronald Choe, Naira Hovakimyan, M Christopher Cotting, Robert B Jeffrey, Margaret P Blackstun, T Paul Fulkerson, Timothy R Lau, and Shawn S Stephens. Evaluation of an $\mathcal{L}_1$ Adaptive Flight Control Law on Calspan's variable-stability Learjet. *AIAA Journal of Guidance, Control, and Dynamics*, 40(4):1051–1060, 2017.

Kong Yao Chee, Thales C Silva, M Ani Hsieh, and George J Pappas. Enhancing sample efficiency and uncertainty compensation in learning-based model predictive control for aerial robots. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9435–9441. IEEE, 2023.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*, 34:15084–15097, 2021.

Yikun Cheng, Pan Zhao, Fanxin Wang, Daniel J Block, and Naira Hovakimyan. Improving the Robustness of Reinforcement Learning Policies with $\mathcal{L}_1$ Adaptive Control. *IEEE Robotics and Automation Letters*, 7(3):6574–6581, 2022.

Yikun Cheng, Pan Zhao, and Naira Hovakimyan. Safe and efficient reinforcement learning using disturbance-observer-based control barrier functions. In *Learning for Dynamics and Control Conference*, pp. 104–115. PMLR, 2023.

Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline rl via supervised learning? *arXiv preprint arXiv:2112.10751*, 2021.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Aditya Gahlawat, Sambhu H Karumanchi, and Naira Hovakimyan. $\mathcal{L}_1$-drac: Distributionally robust adaptive control. *arXiv preprint arXiv:2509.04619*, 2025.

Irene Gregory, Chengyu Cao, Enric Xargay, Naira Hovakimyan, and Xiaotian Zou. $\mathcal{L}_1$ Adaptive Control Design for NASA AirSTAR Flight Test Vehicle. In *AIAA Guidance, Navigation, and Control Conference*, pp. 5738, 2009.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Naira Hovakimyan and Chengyu Cao. $\mathcal{L}_1$ *Adaptive Control Theory: Guaranteed Robustness with Fast Adaptation*. SIAM, 2010.

Kaizhe Hu, Ray Chen Zheng, Yang Gao, and Huazhe Xu. Decision transformer under random frame dropping. *arXiv preprint arXiv:2303.03391*, 2023.

Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE transactions on intelligent transportation systems*, 23(6):4909–4926, 2021.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1179–1191, 2020.

Kuang-Huei Lee, Ofir Nachum, Mengjiao Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, and Igor Mordatch. Multi-game decision transformers. *Advances in Neural Information Processing Systems*, 35, 2022.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, New York, NY, 2nd edition, 2006. ISBN 978-0387-30303-1.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Kishan Panaganti, Zaiyan Xu, Dileep Kalathil, and Mohammad Ghavamzadeh. Robust reinforcement learning using offline data. In *Advances in Neural Information Processing Systems*, volume 35, pp. 32211–32224, 2022.

Keiran Paster, Sheila McIlraith, and Jimmy Ba. You can't count on luck: Why decision transformers and rvs fail in stochastic environments. In *Advances in Neural Information Processing Systems*, volume 35, pp. 38966–38979, 2022.

Jintasit Pravitra, Kasey A Ackerman, Chengyu Cao, Naira Hovakimyan, and Evangelos A Theodorou. $\mathcal{L}_1$-Adaptive MPPI Architecture for Robust and Agile Control of Multirotors. In *IEEE International Conference on Intelligent Robots and Systems*, pp. 7661–7666, 2020.

Marc Rigter, Bruno Lacerda, and Nick Hawes. RAMBO-RL: Robust adversarial model-based offline reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 16082–16097, 2022.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Laixi Shi and Yuejie Chi. Distributionally robust model-based offline reinforcement learning with near-optimal sample complexity. *arXiv preprint arXiv:2208.05767*, 2022.

Ruizhe Shi, Yuyao Liu, Yanjie Ze, Simon Shaolei Du, and Huazhe Xu. Unleashing the power of pre-trained language models for offline reinforcement learning. In *The Twelfth International Conference on Learning Representations*, 2024.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, 2017.

Minjun Sung, Sambhu Harimanas Karumanchi, Aditya Gahlawat, and Naira Hovakimyan. Robust model based reinforcement learning using $\mathcal{L}_1$ adaptive control. In *The Twelfth International Conference on Learning Representations*, 2024.

Xiaohang Tang, Afonso Marques, Parameswaran Kamalaruban, and Ilija Bogunovic. Adversarially robust decision transformer. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.

Fan Wu, Linyi Li, Chejian Xu, Huan Zhang, Bhavya Kailkhura, Krishnaram Kenthapadi, Ding Zhao, and Bo Li. Copa: Certifying robust policies for offline reinforcement learning against poisoning attacks. In *International Conference on Learning Representations*, 2022a.

Yueh-Hua Wu, Xiaolong Wang, and Masashi Hamaya. Elastic decision transformer. *Advances in neural information processing systems*, 36:18532–18550, 2023.

Zhuohuan Wu, Sheng Cheng, Kasey A Ackerman, Aditya Gahlawat, Arun Lakshmanan, Pan Zhao, and Naira Hovakimyan. $\mathcal{L}_1$ Adaptive Augmentation for Geometric Tracking Control of Quadrotors. In *International Conference on Robotics and Automation*, pp. 1329–1336, 2022b.

Jiawei Xu, Rui Yang, Shuang Qiu, Feng Luo, Meng Fang, Baoxiang Wang, and Lei Han. Tackling data corruption in offline reinforcement learning via sequence modeling. In *International Conference on Learning Representations*, 2025.

Taku Yamagata, Ahmed Khalil, and Raul Santos-Rodriguez. Q-learning decision transformer: Leveraging dynamic programming for conditional sequence modelling in offline rl. In *International Conference on Machine Learning*, pp. 38989–39007. PMLR, 2023.

Rui Yang, Chenjia Bai, Xiaoteng Ma, Zhaoran Wang, Chongjie Zhang, and Lei Han. Rorl: Robust offline reinforcement learning via conservative smoothing. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23851–23866, 2022.

Rui Yang, Han Zhong, Jiawei Xu, Amy Zhang, Chongjie Zhang, Lei Han, and Tong Zhang. Towards robust offline reinforcement learning under diverse data corruption. In *The Twelfth International Conference on Learning Representations*, 2024.

Zhihe Yang and Yunjian Xu. DMBP: Diffusion model-based predictor for robust offline reinforcement learning against state observation perturbations. In *The Twelfth International Conference on Learning Representations*, 2024.

Chenlu Ye, Rui Yang, Quanquan Gu, and Tong Zhang. Corruption-robust offline reinforcement learning with general function approximation. *Advances in Neural Information Processing Systems*, 36:36208–36221, 2023.

Xuezhou Zhang, Yiding Chen, Xiaojin Zhu, and Wen Sun. Corruption-robust offline reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 5757–5773. PMLR, 2022.

Qinqing Zheng, Amy Zhang, and Aditya Grover. Online decision transformer. In *International Conference on Machine Learning*, pp. 27042–27059. PMLR, 2022.

A THE AFFINE $\mathcal{L}_1$ BASELINE

For completeness we elaborate on the affine baseline (Algorithm 2), the affinized $\mathcal{L}_1$ correction of Sung et al. (2024) applied to the DT state head. With the locally affine surrogate of equation 3 at the reference action $\bar{a}$ it was last built at, $\hat{f}_\theta^{\text{aff}}(\mathcal{H}_t, a; \bar{a}) := \hat{g}_\theta(\mathcal{H}_t, \bar{a}) + \hat{h}_\theta(\mathcal{H}_t, \bar{a})\, a$, as reference model, and the finite-difference drift $\hat{\varphi}_t^{\text{aff}}(a; \bar{a}) := \big(\hat{f}_\theta^{\text{aff}}(\mathcal{H}_t, a; \bar{a}) - s_t\big)/T_s$ it induces, the state predictor and the piecewise-constant adaptive law can be written as:

$$\dot{\hat{s}}(t) = \hat{\varphi}_t^{\text{aff}}\big(a_t^{\text{cmd}}; \bar{a}\big) + \hat{\sigma}(t) + A_s\, \tilde{s}(t), \qquad \hat{s}(0) = s_0, \tag{12a}$$

$$\hat{\sigma}_t = -\Phi^{-1}\mu_t, \tag{12b}$$

where A_s is Hurwitz, $\tilde{s} := \hat{s} - s$, Φ and μ_t are defined as in Section 2.2. The predictor equation 12a is equation 4 with the surrogate in place of the state head, and the two differ in their drift alone, which is what puts the affinization error $\hat{f}_\theta - \hat{f}_\theta^{\text{aff}}$ into the error dynamics of the affine $\mathcal{L}_1$ baseline. Algorithm 2 advances it by the forward-Euler step of equation 21 with $\hat{f}_\theta^{\text{aff}}$ in place of $\hat{f}_\theta$. Between resets, both predictor updates therefore have the form derived in Section B.1, with their respective drift models.

The matched component $\hat{\sigma}_m$ is the coordinate vector of $\hat{\sigma}$ in the input directions. The surrogate enters equation 12a through the drift $\hat{\varphi}_t^{\text{aff}}$, whose action Jacobian is $\hat{h}_\theta/T_s$, so the split $\hat{\sigma} = (\hat{h}_\theta/T_s)\,\hat{\sigma}_m + \hat{h}_\theta^{\perp}\,\hat{\sigma}_{um}$ of Section 2.2 is taken in that matrix, with $\hat{h}_\theta^{\perp}$ an orthonormal basis of $\text{null}(\hat{h}_\theta^{\top})$. When $\hat{h}_\theta$ has full column rank,

$$\hat{\sigma}_{m,t} = \hat{h}_\theta^{\dagger}\,\hat{d}_t = \big(\hat{h}_\theta^{\top}\hat{h}_\theta\big)^{-1}\hat{h}_\theta^{\top}\,\hat{d}_t, \tag{13}$$

with $\hat{h}_\theta = \hat{h}_\theta(\mathcal{H}_t, \bar{a})$ and the one-step displacement estimate $\hat{d}_t = T_s\hat{\sigma}_t$ of Section 3.1. Filtering $-\hat{\sigma}_{m,t}$ through equation 9 gives the correction δa_t^{cmd}, which is the adaptive input $\tilde{a}$ of Section 2.2. The commanded action is $a_t^{\text{cmd}} = a_t^0 + \delta a_t^{\text{cmd}}$, where δa_t^{cmd} is clipped to make a_t^{cmd} satisfy the actuator bounds. The surrogate coefficients are recomputed at the current context $\mathcal{H}_t$ on every step while holding the reference action fixed. A reference-action refresh occurs when the induced error exceeds ϵ_a; in the switching baseline of Algorithm 2, which follows Sung et al. (2024), this also resets the predictor and correction filter. Their limitations already discuss the tolerance tradeoff. The carry variants in Table 3 retain both states to test the effect of this reset; retaining the states alone has mixed effects on mean return. The reset is a property of this switching construction, not a requirement of affine $\mathcal{L}_1$ compensation in general.

Algorithm 2 Affine $\mathcal{L}_1$ test-time compensation of a DT. (Baseline adapted from Sung et al. (2024))

Input: trained DT $\mathcal{M}_\theta$, environment `env`, context length K, horizon T, target return R_0; sampling interval T_s, observer pole α_s; filter cutoff ω_c; actuator bounds $(a_{\text{lo}}, a_{\text{hi}})$; affinization tolerance ϵ_a
$s_0 \leftarrow$ `env.reset()`; $\tau \leftarrow ()$; $\hat{s}_0 \leftarrow s_0$; $\delta a_{-1}^{\text{cmd}} \leftarrow 0$; $a_{-1}^{\text{cmd}} \leftarrow 0$
for $t = 0, 1, \ldots, T-1$ **do**
  $\mathcal{H}_t \leftarrow (\tau_{t-K+1:t-1}, R_t, s_t)$ ▷ context of Section 2.1
  $a_t^0 \leftarrow \pi_\theta(\mathcal{H}_t)$ ▷ policy head
  if $t = 0$ **or** $\|\hat{f}_\theta(\mathcal{H}_t, a_t^0) - \hat{f}_\theta^{\text{aff}}(\mathcal{H}_t, a_t^0; \bar{a})\|_2 > \epsilon_a$ **then**
    $\bar{a} \leftarrow a_t^0$; $\hat{s}_t \leftarrow s_t$; $\delta a_{t-1}^{\text{cmd}} \leftarrow 0$ ▷ move the reference, restart the estimator and the filter
  end if
  $\hat{f}_\theta^{\text{aff}} \leftarrow$ the surrogate equation 3 at $(\mathcal{H}_t, \bar{a})$ ▷ current context, current reference
  $\tilde{s}_t \leftarrow \hat{s}_t - s_t$; $\hat{\sigma}_t \leftarrow$ adaptive law (equation 12b); $\hat{d}_t \leftarrow T_s\hat{\sigma}_t$
  $\delta a_t \leftarrow -\hat{\sigma}_{m,t}$, minus the matched component of equation 13 ▷ closed form
  Apply low-pass filter to δa_t to get δa_t^{cmd} (equation 9)
  $\delta a_t^{\text{cmd}} \leftarrow \text{clip}(\delta a_t^{\text{cmd}}, a_{\text{lo}} - a_t^0, a_{\text{hi}} - a_t^0)$; $a_t^{\text{cmd}} \leftarrow a_t^0 + \delta a_t^{\text{cmd}}$
  $s_{t+1}, r_t, \texttt{done} \leftarrow$ `env.step`(a_t^{cmd})
  $R_{t+1} \leftarrow R_t - r_t$ ▷ DT return-to-go update
  $\hat{s}_{t+1} \leftarrow \hat{f}_\theta^{\text{aff}}(\mathcal{H}_t, a_t^{\text{cmd}}; \bar{a}) + (I + T_sA_s)\tilde{s}_t + T_s\hat{\sigma}_t$; extend τ by $(R_t, s_t, a_t^{\text{cmd}})$
  if `done` **then break**
end for

B Extended Analysis

B.1 State predictor and adaptive law for a non-control-affine model

This subsection is an extended description of the state predictor and adaptive law for a non-control-affine model. The point it makes precise is that the two estimation components of $\mathcal{L}_1$AC never depend on an input matrix, so nothing in them requires the model to be affine in the action. The affinization of equation 3 is needed only by the closed-form cancellation, which Section 3.2 replaces.

Consider a system

$$\dot{s}(t) = f\big(s(t), a(t)\big) + d\big(t, s(t), a(t)\big), \tag{14}$$

with state $s \in \mathbb{R}^n$, action $a \in \mathbb{R}^m$, a known nominal drift $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$, and an unknown perturbation $d(t, s(t), a(t)) \in \mathbb{R}^n$. The drift is an arbitrary nonlinear function of both arguments. The perturbation carries everything the nominal drift does not, including exogenous disturbance, unmodelled dynamics, and so on. In this work, f is instantiated as the finite-difference drift of the DT state head (Section 3.1), but that instance plays no part in what follows.

With $\hat{s}(0) = s_0$, prediction error $\tilde{s} := \hat{s} - s$, and A_s Hurwitz, the predictor is

$$\dot{\hat{s}}(t) = f\big(s(t), a(t)\big) + \hat{\sigma}(t) + A_s\, \tilde{s}(t), \tag{15}$$

which is equation 4 for the drift of Section 3.1. Note that the first term is the nonlinear drift itself and not an affine surrogate of it. And it is evaluated at the system state and the applied action, so it is the same quantity that drives the plant in equation 14.

Subtracting equation 14 from equation 15,

$$\dot{\tilde{s}}(t) = A_s\, \tilde{s}(t) + \Big(\hat{\sigma}(t) - d\big(t, s(t), a(t)\big)\Big), \qquad \tilde{s}(0) = 0. \tag{16}$$

The drift has cancelled identically, which explains why the estimator needs no control-affine structure. Had the predictor used a surrogate, the gap between the two vector fields would survive as an extra forcing term in equation 16, indistinguishable there from the perturbation it is trying to estimate; bounding that term is what the tolerance ϵ_a and the switching law of the affine construction exist for. What is left is linear and time-invariant in $\tilde{s}$, exponentially stable because A_s is Hurwitz, and driven by the estimation error alone. So $\tilde{s}$ measures how far $\hat{\sigma}$ is from $d(t, s(t), a(t))$, and it is the signal the adaptive law needs.

We now derive the piecewise-constant adaptive law. Sample at $t_i := iT_s$. Hold the estimate constant on each interval, $\hat{\sigma}(t) = \hat{\sigma}_i$ for $t \in [t_i, t_{i+1})$, and integrate the linear system equation 16 over that interval:

$$\begin{aligned} \tilde{s}(t_{i+1}) = e^{A_s T_s}\, \tilde{s}(t_i) + \bigg(\int_{t_i}^{t_{i+1}} e^{A_s(t_{i+1}-\tau)}\, \mathrm{d}\tau\bigg)\hat{\sigma}_i \\ - \int_{t_i}^{t_{i+1}} e^{A_s(t_{i+1}-\tau)}\, d\big(\tau, s(\tau), a(\tau)\big)\, \mathrm{d}\tau. \end{aligned} \tag{17}$$

The middle integral evaluates in closed form because the integrand no longer involves the state, which gives

$$\tilde{s}(t_{i+1}) = e^{A_s T_s}\, \tilde{s}(t_i) + \Phi(T_s)\, \hat{\sigma}_i + R_{i+1}, \qquad \Phi(T_s) = A_s^{-1}\big(e^{A_s T_s} - I\big), \tag{18}$$

where

$$R_{i+1} := -\int_{t_i}^{t_{i+1}} e^{A_s(t_{i+1}-\tau)}\, d\big(\tau, s(\tau), a(\tau)\big)\, \mathrm{d}\tau \tag{19}$$

collects the uncertainty that entered during the interval. Of the three terms on the right of equation 18, the first two are known at t_i, since $\tilde{s}(t_i)$ is measured and $\hat{\sigma}_i$ is up to our choice, and the third is not known at t_i and cannot be. The piecewise-constant law is the choice that annihilates the known part,

$$\hat{\sigma}_i = -\Phi^{-1}(T_s)\, e^{A_s T_s}\, \tilde{s}(t_i), \qquad \hat{\sigma}_0 = 0, \tag{20}$$

which is equation 5. It collapses equation 18 to $\tilde{s}(t_{i+1}) = R_{i+1}$: at every sample the prediction error is exactly the residual accumulated over the preceding interval, so the law feeds back a one-interval-delayed measurement of $d(t, s(t), a(t))$ and assumes nothing about it beyond integrability. The price

is that delay. It is exactly one sampling interval, so a shorter T_s shortens it, but $\Phi^{-1}(T_s)$ grows as T_s falls, and managing that tradeoff is what the filter of Section 3.3 is for.

The cancellation above is a continuous-time statement: it holds for equation 16 integrated exactly over the interval. Both algorithms instead advance their predictor by one forward-Euler step of length T_s. Because the drift is a finite difference about s_t, the step telescopes to

$$\hat{s}_{t+1} = \hat{f}_\theta\big(\mathcal{H}_t, a_t^{\mathrm{cmd}}\big) + (I + T_s A_s)\,\tilde{s}_t + T_s\,\hat{\sigma}_t, \tag{21}$$

which is Algorithm 1's predictor line. Subtracting s_{t+1} from it and substituting the law equation 20 leaves

$$\tilde{s}_{t+1} = \hat{f}_\theta\big(\mathcal{H}_t, a_t^{\mathrm{cmd}}\big) - s_{t+1} + M\,\tilde{s}_t, \qquad M = (I + T_s A_s) - T_s\,\Phi^{-1}(T_s)\,e^{A_s T_s}. \tag{22}$$

The forcing term is the model's realized one-step error at the commanded action, playing a role analogous to R_{i+1} in equation 18. The exact continuous-time interval solution retains that disturbance integral, while the adaptive law cancels the contribution of the preceding prediction error. The Euler recurrence additionally retains $M\tilde{s}_t$. For $A_s = \alpha_s I$ every matrix is a multiple of the identity and

$$M = \left(1 + \alpha_s T_s - \frac{\alpha_s T_s\, e^{\alpha_s T_s}}{e^{\alpha_s T_s} - 1}\right) I = \Big(\tfrac{1}{2}\alpha_s T_s + O\big((\alpha_s T_s)^2\big)\Big) I, \tag{23}$$

which at the settings of Table 5 gives $M \approx -0.0252 I$ on halfcheetah and $M \approx -0.0040 I$ on hopper and walker2d.

B.2 Proofs for Section 4

The propositions consider the ideal Gauss–Newton quadratic model and the affine inversion objective; their identification with an implemented update requires the stated solver conditions.

Proposition 1 (the first step). Write $H := J_{a,t}^\top Q J_{a,t} + \lambda R$. Then $J_{a,t}^\top Q J_{a,t} \succeq 0$ because $Q \succ 0$, and $\lambda R \succ 0$ because $\lambda > 0$ and $R \succ 0$, so $H \succ 0$. This is the Hessian of the quadratic q_t, which is therefore strictly convex, so its unique stationary point is its global minimizer. Its gradient $\nabla q_t(\delta a) = H\,\delta a + J_{a,t}^\top Q\,\hat{d}_t$ vanishes only at $\delta a_t^{\mathrm{gn}} = -H^{-1} J_{a,t}^\top Q\,\hat{d}_t$. For the identification with the solve: at $\delta a^{(0)} = 0$ we have $J_r = J_{a,t}$ and $r_t(0) = \hat{d}_t$, hence $\nabla J_t(0) = J_{a,t}^\top Q\,\hat{d}_t$, and equation 8 with $\mu_0 = 0$ reads $H\,p^{(0)} = -J_{a,t}^\top Q\,\hat{d}_t$, so the unit-length update is $\delta a^{(1)} = p^{(0)} = \delta a_t^{\mathrm{gn}}$. □

Proposition 2 (exact reduction). Control-affineness gives $\Delta\hat{f}_t(\delta a) = \big(\hat{g}_\theta + \hat{h}_t(a_t^0 + \delta a)\big) - \big(\hat{g}_\theta + \hat{h}_t a_t^0\big) = \hat{h}_t\,\delta a$ for every $\delta a \in \mathbb{R}^m$, so at $Q = I$ and $\lambda = 0$, $J_t(\delta a) = \frac{1}{2}\|\hat{d}_t + \hat{h}_t \delta a\|^2$. Full column rank makes $\hat{h}_t^\top \hat{h}_t \succ 0$, so J_t is strictly convex and its unique stationary point $\delta a_t^\star = -(\hat{h}_t^\top \hat{h}_t)^{-1}\hat{h}_t^\top \hat{d}_t$ is the minimizer. For the identification with $\hat{\sigma}_{m,t}$, write $\hat{d}_t = \hat{h}_t \hat{\sigma}_{m,t} + \hat{h}_t^\perp \hat{\sigma}_{um,t}$, which is unique because $[\hat{h}_t,\ \hat{h}_t^\perp]$ is a basis of $\mathbb{R}^n$. The columns of $\hat{h}_t^\perp$ span the orthogonal complement of $\mathrm{range}(\hat{h}_t)$, so $\hat{h}_t^\top \hat{h}_t^\perp = 0$ and $\hat{h}_t^\top \hat{d}_t = \hat{h}_t^\top \hat{h}_t\,\hat{\sigma}_{m,t}$, giving $(\hat{h}_t^\top \hat{h}_t)^{-1}\hat{h}_t^\top \hat{d}_t = \hat{\sigma}_{m,t}$ and hence equation 10. □

C Extended Experiments

C.1 Experimental details

We use D4RL datasets (Fu et al., 2020) for HalfCheetah, Hopper, and Walker2d. Table 4 lists the DT default hyperparameters, Tables 5, 6 and 8 those of the direct and affine $\mathcal{L}_1$ compensators. A value that differs by environment is given as HalfCheetah / Hopper / Walker2d. Three random seeds are used. And we evaluate each cell on three checkpoints with ten episodes each. The action disturbance is a random walk, $\varepsilon^a_{t+\Delta t} = \varepsilon^a_t + \xi_t$ with $\varepsilon^a_0 = 0$ and $\xi_t \sim \mathcal{N}(0,\, k^2\sigma^2\Delta t\, I)$, where Δt is the environment's control interval and k the magnitude scale of Figure 2. Table 7 gives the checkpoints and σ per cell. The per-step times of Table 2 are the controller's wall time between the return of one simulator call and the entry of the next, measured by an NVIDIA RTX 4080.

Table 4: Hyperparameters of the DT.

Hyperparameter	**Value**	**Hyperparameter**	**Value**
Context length K	20	Weight decay	1×10^{-4}
Batch size	64	Grad norm clip	0.25
Embedding dimension	128	Warmup steps	10000
Number of layers	3	Steps per iteration	1000
Number of attention heads	1	State loss weight μ	0.2
Activation function	`ReLU`	RTG loss weight ν	0.005
Dropout	0.1	Max episode length T	1000
Learning rate	1×10^{-4}	Target return R_0	12000 / 3600 / 5000

Table 5: Hyperparameters of state predictor and low-pass filter, shared by direct and affine $\mathcal{L}_1$.

Symbol	**Meaning**	**Value**
α_s	observer pole ($A_s = \alpha_s I$, Hurwitz)	-1
T_s	sampling interval (matched to Δt)	50 / 8 / 8 ms
ω_c	cutoff frequency (rad/s)	8.6 / 1.3 / 1.3

Table 6: Settings of direct $\mathcal{L}_1$ compensator (Algorithm 1). The regularizer weight λ is in Table 8.

Symbol	**Meaning**	**Value**
Q	residual weight	I
R	correction regularizer metric	I
K_{it}	accepted-iteration cap	1
μ_0	initial damping ($\times 10$ reject, $\times 0.3$ accept)	10^{-3}
line-search steps	backtracking ladder length	8
Armijo c	sufficient-decrease constant	10^{-4}

Table 7: Checkpoints and σ of each cell.

Env	Dataset	checkpoints	σ
halfcheetah	expert	$50, 60, 70$	0.035
	medium	$65, 70, 75$	0.0875
	medium-replay	$65, 70, 75$	0.035
hopper	expert	$580, 590, 600$	0.045
	medium	$65, 70, 75$	0.225
	medium-replay	$90, 95, 100$	0.045
walker2d	expert	$80, 90, 99$	0.09
	medium	$65, 70, 75$	0.09
	medium-replay	$65, 70, 75$	0.09

Table 8: The affine baseline's affinization tolerance ϵ_a and the regularizer weight λ of the direct $\mathcal{L}_1$, which in Table 3 is also the weight of its two regularized affine arms.

Setting	Env	Dataset	ϵ_a	λ
Table 1	halfcheetah	expert	2	3
		medium	0.01	0.03
		medium-replay	0.1	3
	hopper	expert	0.02	3
		medium	0.2	3
		medium-replay	0.4	1.5
	walker2d	expert	0.1	3
		medium	0.5	3
		medium-replay	0.1	3
Table 2, Figure 2	halfcheetah	expert	0.05	3
	hopper	expert	0.02	3
	walker2d	expert	0.1	3
Table 3	halfcheetah	expert	2	0.3
	hopper	medium-replay	1	0.3
	walker2d	medium	4	0.3

C.2 Figure 2 in tabular form

Table 9: Magnitude sweep in tabular form.

Env	k	DT	Affine $\mathcal{L}_1$	Direct $\mathcal{L}_1$
halfcheetah	0.0	92.5 ± 0.7	92.6 ± 0.8	92.1 ± 1.6
	0.5	85.1 ± 1.4	84.9 ± 1.6	85.8 ± 1.2
	1.0	66.1 ± 2.4	65.7 ± 4.1	72.1 ± 1.8
	1.5	48.8 ± 3.6	48.8 ± 3.6	56.0 ± 3.3
	2.0	37.1 ± 2.0	37.3 ± 2.0	43.3 ± 3.0
hopper	0.0	109.3 ± 3.1	110.7 ± 0.6	110.6 ± 1.1
	0.5	100.0 ± 3.5	99.9 ± 3.7	101.6 ± 7.4
	1.0	85.9 ± 5.0	81.4 ± 8.3	88.1 ± 7.5
	1.5	66.4 ± 5.7	66.5 ± 6.6	72.1 ± 8.8
	2.0	61.2 ± 5.1	60.8 ± 7.0	64.3 ± 8.4
walker2d	0.0	108.4 ± 0.6	108.5 ± 0.7	108.4 ± 0.7
	0.5	103.6 ± 2.9	102.8 ± 2.9	103.3 ± 5.1
	1.0	78.7 ± 7.4	80.1 ± 8.4	83.1 ± 6.1
	1.5	47.9 ± 8.4	51.1 ± 4.9	56.1 ± 8.7
	2.0	30.0 ± 6.3	30.0 ± 7.0	32.8 ± 3.7

Table 10: Decomposing the gap between the equipped affine baseline and direct $\mathcal{L}_1$, on the three cells of Table 3, whose last two rows are each block's first and last. Each arm adds one ingredient to the arm above it. +*reference* sets the affinization tolerance to 10^{-6}, so the refresh fires at every step and the linearization action is the current nominal one. +*state head* gives the predictor the state head evaluated at the applied command in place of its first-order expansion, and takes the undamped ridge step of Proposition 1. +*damping* adds the Levenberg–Marquardt damping and the Armijo ladder of the LM solver.

Env-dataset	arm	parallel	orthogonal	return
halfcheetah expert, $k = 1$	+carry +regularizer	0.45	0.54	68.4
	+reference	0.41	0.43	72.6
	+state head	0.43	0.39	73.8
	+damping	0.44	0.36	73.0
hopper medium-replay, $k = 2$	+carry +regularizer	0.22	0.25	45.9
	+reference	0.22	0.22	48.5
	+state head	0.22	0.22	49.6
	+damping	0.22	0.22	50.0
walker2d medium, $k = 1.5$	+carry +regularizer	0.13	0.26	43.4
	+reference	0.15	0.16	44.5
	+state head	0.15	0.15	45.2
	+damping	0.15	0.16	45.2

C.3 Decomposing the gap between the equipped baseline and direct $\mathcal{L}_1$

Table 3's last two rows differ in three ways, and Table 10 removes them one at a time.

The action reference. Setting ϵ_a to 10^{-6} drives the measured refresh rate to 1.00, and with the estimator carried a refresh replaces f, g and $g^{\perp}$ at the current nominal action while the predictor and filter states keep running. It recovers $+4.2$, $+2.6$ and $+1.2$ of the gap. This is the largest of the three differences on every cell, and the only one that moves the diagnostics appreciably: the orthogonal share falls by 0.11, 0.03 and 0.10, which is 61, 100 and 100 percent of each cell's total fall, while the parallel share moves by at most 0.04.

The drift model. The predictor's drift becomes the state head evaluated at the applied command, in place of the surrogate evaluated there, which changes the displacement estimate the solve is handed. The change adds $+1.2$, $+1.2$ and $+0.7$ to the return. And only in the halfcheetah cell it causes a noticeable shift in the parallel and orthogonal metrics. Therefore we report it as a secondary effect.

Damping and backtracking. Adding damping and backtracing back moves mean return by -0.8, $+0.4$ and $+0.0$ and causes no noticeable change in the parallel and orthogonal metrics. At these operating points they rarely bind.

C.4 Nominal robustness, and effect of the state-loss weight

This subsection asks three questions on walker2d and hopper expert at $\nu = 0.005$, $\mu \in \{0, 0.2, 0.4, 0.8, 2.0\}$: how the *nominal* DT's robustness evolves during training, whether μ changes it at convergence, and how μ affects the disturbance compensation.

Robustness keeps rising after clean return has plateaued. On walker2d the clean ($k = 0$) return of every μ plateaus near 108 by iteration 30, yet return under noise at $k = 1$ keeps climbing to its own plateau near iteration 85 to 95. Hopper's clean return plateaus by iteration 100, but return under noise at $k = 1$ plateaus after iteration 500 (Figure 3). Every μ then climbs from 62 to 70 at iteration 100 into a common band of 80 to 82.

At convergence, μ does not change nominal robustness. Once each μ is trained to its own plateau, nominal robustness is flat in μ on both plants (Figure 4): from $\mu = 0$ to $\mu = 2.0$ walker2d moves 79.2 to 79.0 at $k = 1$ and 47.0 to 48.6 at $k = 1.5$, hopper 79.5 to 79.7 and 66.2 to 66.7.

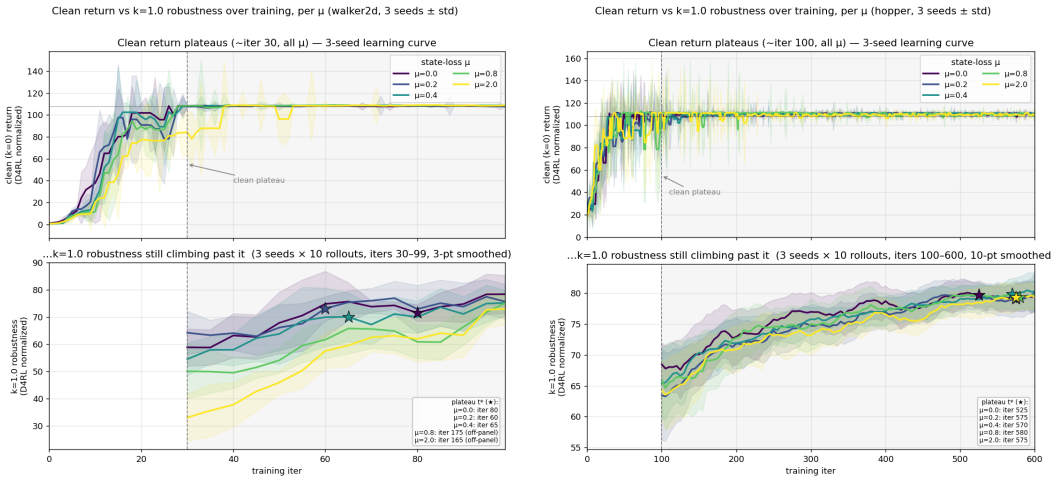

Figure 3: Clean return (upper row of each panel) and return under $k = 1$ noise (lower row) against training iteration, one curve per μ, on walker2d (left, to iteration 100) and hopper (right, to 600), robustness shown from each plant's clean-return plateau onward. Stars mark the detected robustness plateau. Robustness reaches its own plateau well after clean return does.

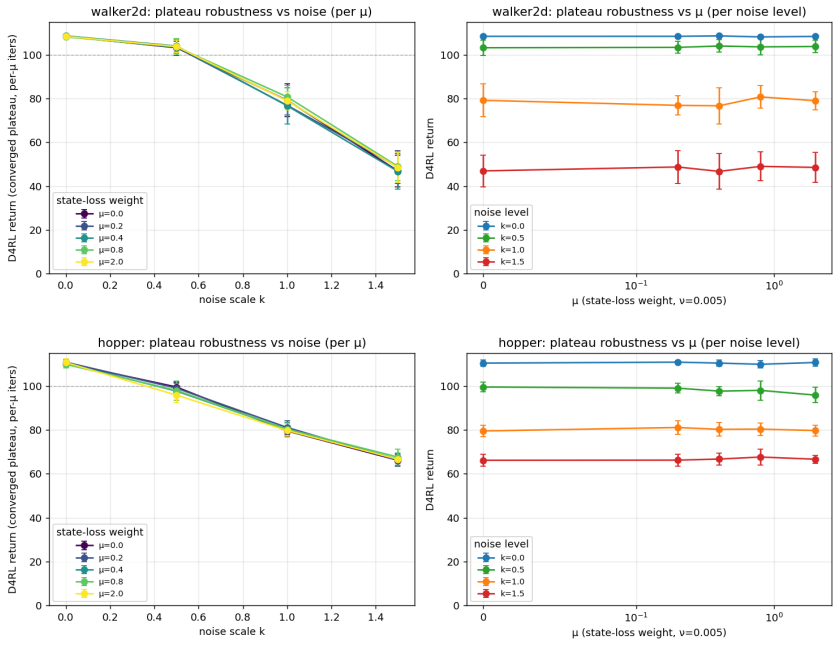

Figure 4: Convergence-matched nominal robustness on walker2d (top row) and hopper (bottom row). Left: D4RL return against noise scale k, one curve per μ. Right: the same returns against μ, one curve per noise scale.

μ gates the correction rather than scaling it. With no next-state loss at all the head supports no useful correction. Above zero the tested weights are indistinguishable: on walker2d the benefit pooled over $k \in \{0.5, 1.0, 1.5\}$ runs from $+9.9$ at $\mu = 0.2$ down to $+7.6$ at $\mu = 2.0$, against entry standard deviations near 8; on hopper the weights are equally indistinguishable, at smaller benefits of $+0.5$ to $+2.8$ (Figure 5). These runs show no sensitivity of the correction to μ once the state head is trained at all.

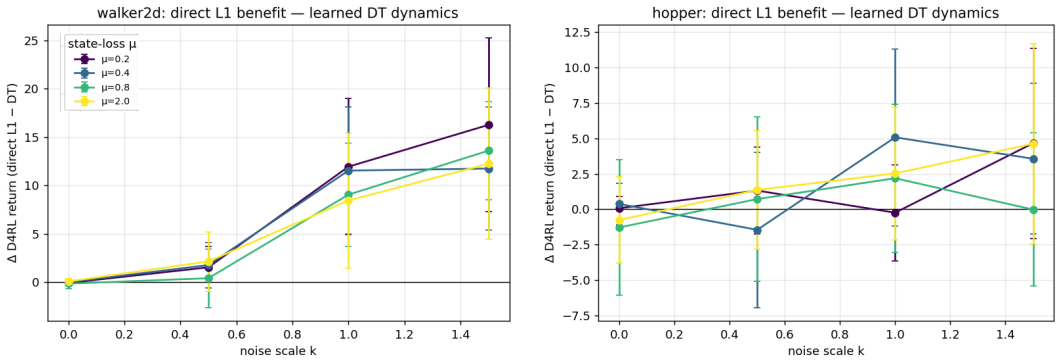

Figure 5: Benefit of direct $\mathcal{L}_1$ over the nominal DT against noise scale k, one curve per μ, on walker2d (left) and hopper (right).