BEYOND FORECAST ACCURACY: EVALUATING NEURAL OPERATORS FOR DIFFERENTIABLE PDE CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural operators learn mappings between function spaces and are inexpensive to evaluate and differentiate, making them a natural choice of surrogate for controlling systems governed by partial differential equations (PDEs), where repeated numerical solves render online optimization expensive. Differentiable predictive control (DPC) exploits both properties: it trains a neural feedback policy offline by differentiating a control objective through surrogate rollouts, so deployment reduces to a single forward pass. The surrogate thus plays a dual role, predicting the consequences of actions and supplying the sensitivities that drive policy updates. Our framework trains a neural-operator surrogate on open-loop trajectories, freezes it, optimizes the policy through predicted closed-loop rollouts, and deploys it on the numerical solver. We compare Fourier, Recurrent, and Time-integrated DeepONet operators with single- and multi-step training on stabilization across linear, nonlinear, dispersive, and chaotic PDEs. We find that forecast accuracy alone is an insufficient proxy for controller quality: surrogates with comparable prediction errors can yield policies that differ by orders of magnitude in terminal tracking error. A policy-dependent transfer bound and a counterexample in which vanishing forward-model error coexists with persistent policy-gradient error formalize why this can occur. With multi-step training, DPC outperforms model-free reinforcement learning while avoiding the online cost of model predictive control.

1 INTRODUCTION

Physical phenomena such as heat transfer, wave propagation, turbulence, phase separation, and interface dynamics are modeled by partial differential equations (PDEs) (Evans, 2010; Pope, 2000). Controlling such systems, whether stabilizing a turbulent flow or steering a temperature profile to a target, is naturally posed as PDE-constrained optimization (Hinze et al., 2009; Tröltzsch, 2010; Kim & Bewley, 2007; Brunton & Noack, 2015). Because PDE states are infinite-dimensional fields, they are computed through spatial discretizations (LeVeque, 2007; Trefethen, 2000) whose dimension can reach millions of degrees of freedom (Benner et al., 2015), so every evaluation of a candidate control requires an expensive numerical solve. Classical control theory offers rigorous frameworks, but these methods typically demand accurate system models, computationally intensive online optimization, or problem-specific designs that do not transfer across domains (Krstic & Smyshlyaev, 2008; Kouvaritakis & Cannon, 2016).

Learning-based approaches offer an alternative by approximating control policies from data (Farahmand et al., 2017; Rabault et al., 2019; Bhan et al., 2024; Zhang et al., 2024; Becktepe et al., 2026). Model-free reinforcement learning (Schulman et al., 2017) avoids an explicit model during policy optimization, yet in practice it relies on a simulator for the large number of interactions it requires, so the numerical model is used implicitly and repeatedly (Werner & Peitz, 2023). Model-based approaches instead exploit the model directly (Deisenroth & Rasmussen, 2011; Janner et al., 2019). Among these, differentiable predictive control (DPC) (Drgoňa et al., 2022) is well suited to PDEs (Sarkar et al., 2025; Zanotta et al., 2026): it retains the MPC objective and constraints but moves the optimization offline, training a neural feedback policy with analytic gradients through model rollouts rather than sampled reward estimates, so deployment replaces an online high-dimensional solve with a single forward pass. This requires a differentiable model of the dynamics, which for PDEs has been obtained either by differentiating through numerical solvers (Holl et al., 2020; Zanotta et al., 2026) or by learning a neural-operator surrogate from data (Sarkar et al., 2025).

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

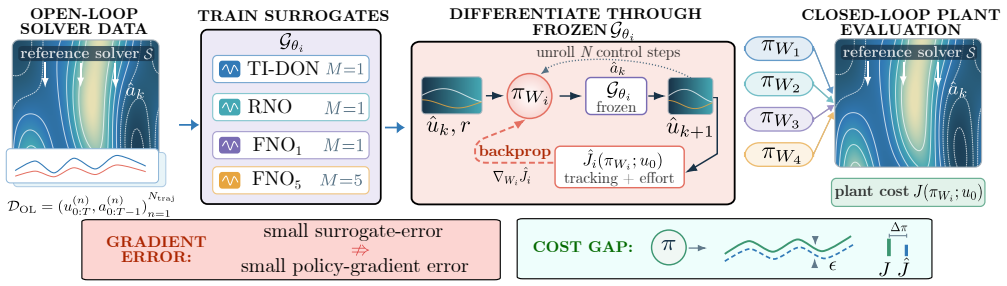

Figure 1: **Overview.** Neural-operator surrogates are trained on open-loop solver data and frozen. A feedback policy is then optimized through each surrogate by DPC and evaluated on the numerical solver. Bottom: accurate predictions bound the cost gap for a fixed policy (Prop. 1), but not the policy-gradient error (Prop. 2).

The surrogate route is attractive because neural operators are cheap to query and differentiate, and a growing family of architectures and training objectives is available (Li et al., 2021; Lu et al., 2021; Liu-Schiaffini et al., 2023; Nayak & Goswami, 2025; Kovachki et al., 2023). Existing benchmarks rank these models by open-loop forecast error (Takamoto et al., 2022; Lippe et al., 2023), and that ranking is often implicitly assumed to carry over to control. In DPC, however, the policy learns through the surrogate's Jacobians and along the closed-loop states it visits, neither of which forecasting measures. Model-based learning has long noted that better prediction need not yield better decisions (Farahmand et al., 2017; Lambert et al., 2020), but these analyses concern models used for planning or value estimation, not policies trained through a surrogate's derivatives. We test this assumption directly: *does the surrogate that forecasts best also train the best controller?* Figure 1 summarizes our approach and findings.

Contributions. We study neural operators not as predictors, but as differentiable models used to train feedback controllers. Our contributions are:

(1) A control-centric evaluation framework for neural operator surrogates. We separate three quantities that are conflated when learned models are used for control, namely open-loop forecast accuracy, surrogate-side closed-loop performance, and plant-side closed-loop performance, and compare Fourier, Recurrent, and Time-integrated DeepONet surrogates, with single- and multi-step training, as frozen plug-ins to one DPC pipeline under identical data and objectives (Section 3).

(2) A theoretical characterization of what forecast accuracy can and cannot certify. We show that surrogate-to-plant transfer is governed by model error along the policy's own closed-loop trajectory rather than by open-loop validation error, and that arbitrarily small forward-model error can coexist with persistent error in the action Jacobian that supplies the policy gradient (Section 4).

(3) An empirical characterization of neural operators for differentiable PDE control. On six `controlgym` benchmarks (Zhang et al., 2024) spanning linear, nonlinear, dispersive, and chaotic dynamics, forecasting and control rankings frequently disagree, comparable forecast errors yield controllers differing by orders of magnitude, and surrogates misestimate their own policies by up to two orders of magnitude. Five-step FNO training is the most consistent configuration, and comparisons with PPO, online MPC, and shifted initial conditions give practical guidance for selecting surrogates for PDE control (Sections 5.1 to 5.4).

2 Related Work

Classical PDE control. Classical methods trade rigor against real-time feasibility. MPC requires an online solve at every step (Kouvaritakis & Cannon, 2016), adjoint methods scale poorly with state dimension (Gunzburger, 2003; Hinze et al., 2009), and backstepping requires plant-specific kernel equations (Krstic & Smyshlyaev, 2008). Neural operators accelerate those kernel computations while retaining stability guarantees (Bhan et al., 2023; 2024; Krstic et al., 2024; Qi et al., 2024), but remain tied to boundary actuation and to designs that must be re-derived for each class of nonlinear PDE.

Reinforcement learning for PDE control. Model-free reinforcement learning has stabilized chaotic dynamics, handled distributed actuation, and discovered flow-control strategies (Bucci et al., 2019; Pan et al., 2018; Rabault et al., 2019), with learned models reducing its sample cost (Zeng et al., 2022; Werner & Peitz, 2023) and benchmarks enabling comparison (Zhang et al., 2024; Becktepe et al., 2026). Its policy gradients are estimated from sampled returns, however, inheriting high variance and sensitivity to reward design, which motivates backpropagating through differentiable dynamics instead.

Differentiable and learned models for control. Holl et al. (2020) computed policy gradients through a differentiable PDE solver, and follow-up work trained solver corrections and flow controllers the same way (Um et al., 2020; Holl & Thuerey, 2024). Differentiable MPC and DPC amortize constrained optimal control into explicit policies (Amos et al., 2018; Drgoňa et al., 2022; 2024). For PDEs, recent methods jointly train operators and policies with physics-informed losses (Güven et al., 2025), reuse such operators across objectives (Lundqvist & Oliveira, 2025), or parameterize the policy as a neural operator (Zanotta et al., 2026); all require the governing equations. Neural operators (Lu et al., 2021; Goswami et al., 2023; Kontolati et al., 2024) also serve as prediction models in learning-based MPC (Hwang et al., 2022; de Jong et al., 2025; Li et al., 2025; Zhao et al., 2025), which retains an online solve at every step. Closest to our work, Sarkar et al. (2025) learn a TI-DeepONet surrogate from data alone, but study one architecture with one-step supervision on diffusion-dominated systems.

Sim-to-real transfer and model mismatch. Absorbing surrogate error into a policy during training is an instance of the sim-to-real gap, which robotics narrows through domain and dynamics randomization (Tobin et al., 2017; Peng et al., 2018; Muratore et al., 2022) and which also arises in operator learning under distribution shift (Goswami et al., 2022). Model-based reinforcement learning limits model exploitation with ensembles and short rollouts (Chua et al., 2018; Janner et al., 2019), while value-aware and Sobolev losses change what the model is fit to (Farahmand et al., 2017; Czarnecki et al., 2017; Lambert et al., 2020). These remedies target robustness to model error rather than the properties a surrogate must have for differentiable policy training, which is what we characterize and measure.

3 Methodology: DPC with Neural-Operator Surrogates

We use the DPC pipeline of Sarkar et al. (2025), summarized in Figure 6: a surrogate is trained on open-loop trajectories, frozen, and used to train a feedback policy that is then deployed on the numerical solver. Algorithms 1 and 2 specify the two training stages, Appendix B gives the general parameterized formulation, and Table 3 lists all symbols.

Control problem. A scalar field $u(x,t)$ on the periodic domain $\Omega = [0, L]$ evolves as $\partial_t u = \mathcal{F}(u) + \sum_{q=1}^{n_a} \Phi_q(x)\, a_q(t)$, where $\mathcal{F}$ is the uncontrolled spatial differential operator and $\Phi_q : \Omega \to \mathbb{R}$ is the profile of actuator q. We discretize Ω with n_x grid points and time with step Δt, hold the amplitudes $\boldsymbol{a}_k$ constant on $[k\Delta t, (k+1)\Delta t)$, and write the numerical solver, which we call the plant, as $\boldsymbol{u}_{k+1} = \mathcal{S}(\boldsymbol{u}_k, \boldsymbol{a}_k)$ with state $\boldsymbol{u}_k \in \mathbb{R}^{n_x}$. Over a horizon of N steps, we seek the parameters $\boldsymbol{W} \in \mathbb{R}^{n_W}$ of a feedback policy $\pi_{\boldsymbol{W}}$ that minimize the expected cost of tracking a fixed reference $\boldsymbol{r} \in \mathbb{R}^{n_x}$:

$$\min_{\boldsymbol{W}} \quad \mathcal{J}(\boldsymbol{W}) = \mathbb{E}_{\boldsymbol{u}_0 \sim \mathcal{D}_{\mathrm{DPC}}}\big[J(\pi_{\boldsymbol{W}}; \boldsymbol{u}_0)\big], \qquad J(\pi; \boldsymbol{u}_0) = \frac{1}{N}\sum_{k=0}^{N-1} \ell(\boldsymbol{u}_{k+1}, \boldsymbol{a}_k), \tag{1a}$$

$$\text{s.t.} \quad \boldsymbol{u}_{k+1} = \mathcal{S}(\boldsymbol{u}_k, \boldsymbol{a}_k), \qquad \boldsymbol{a}_k = \pi_{\boldsymbol{W}}(\boldsymbol{u}_k, \boldsymbol{r}) \in \mathbb{A}, \qquad k = 0, \dots, N-1. \tag{1b}$$

The stage cost is $\ell(\boldsymbol{u}, \boldsymbol{a}) = (\boldsymbol{u} - \boldsymbol{r})^\top \boldsymbol{Q}(\boldsymbol{u} - \boldsymbol{r}) + \boldsymbol{a}^\top \boldsymbol{R}\boldsymbol{a}$, with $\boldsymbol{Q} \in \mathbb{R}^{n_x \times n_x}$ and $\boldsymbol{R} \in \mathbb{R}^{n_a \times n_a}$ weighting tracking error and control effort, and $\mathcal{D}_{\mathrm{DPC}}$ is the distribution of initial states used for policy training. The policy's output transformation enforces the actuator box $\mathbb{A} = \{\boldsymbol{a} : \|\boldsymbol{a}\|_\infty \le a_{\max}\}$ by construction. Appendix B gives the general formulation with PDE parameters, terminal costs, and state constraints.

Surrogate training. A surrogate $\mathcal{G}_{\boldsymbol{\theta}} : \mathbb{R}^{n_x} \times \mathbb{A} \to \mathbb{R}^{n_x}$ with parameters $\boldsymbol{\theta} \in \mathbb{R}^{n_\theta}$ approximates $\mathcal{S}$. We compare FNO, RNO, and TI-DeepONet surrogates (Appendix C). All are trained on the same

dataset $\mathcal{D}_{\mathrm{dyn}}$ of open-loop plant trajectories $\{(\boldsymbol{u}_{0:N}^{(i)}, \boldsymbol{a}_{0:N-1}^{(i)})\}$, generated with random actuation that is independent of the control objective (Appendix E). A training window is a pair (i, k) covering steps k to $k+M$ of trajectory i, where M is the supervision horizon. Starting from $\hat{\boldsymbol{u}}_k^{(i)} = \boldsymbol{u}_k^{(i)}$, we roll the surrogate forward with the recorded actions,

$$\hat{\boldsymbol{u}}_{k+n+1}^{(i)} = \mathcal{G}_{\boldsymbol{\theta}}(\hat{\boldsymbol{u}}_{k+n}^{(i)}, \boldsymbol{a}_{k+n}^{(i)}), \qquad n = 0, \dots, M-1,$$

and minimize the relative error at every step of the window,

$$\mathcal{L}_{\mathrm{dyn}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}|} \sum_{(i,k)\in\mathcal{B}} \sum_{n=1}^{M} \frac{\|\hat{\boldsymbol{u}}_{k+n}^{(i)} - \boldsymbol{u}_{k+n}^{(i)}\|_2}{\|\boldsymbol{u}_{k+n}^{(i)}\|_2}, \tag{2}$$

where $\mathcal{B}$ is a minibatch of windows. We use $M = 1$ for all architectures and additionally $M = 5$ for FNO, denoted FNO_1 and FNO_5.

Policy training through the frozen surrogate. With $\boldsymbol{\theta}$ fixed, the surrogate closed loop from $\boldsymbol{u}_0 \sim \mathcal{D}_{\mathrm{DPC}}$ is

$$\hat{\boldsymbol{a}}_k = \pi_{\boldsymbol{W}}(\hat{\boldsymbol{u}}_k, \boldsymbol{r}), \qquad \hat{\boldsymbol{u}}_{k+1} = \mathcal{G}_{\boldsymbol{\theta}}(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k), \qquad \hat{\boldsymbol{u}}_0 = \boldsymbol{u}_0, \tag{3}$$

and $\widehat{J}(\pi_{\boldsymbol{W}}; \boldsymbol{u}_0)$ denotes Equation 1 evaluated on $(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k)$. We minimize the minibatch average of $\widehat{J}$ over $\boldsymbol{W}$ by gradient descent and denote the result $\boldsymbol{W}^\star$; the reported runs add an intermediate-state tracking term (Appendix C.2). Differentiating Equation 3, the state sensitivity $\boldsymbol{U}_k = \mathrm{d}\hat{\boldsymbol{u}}_k/\mathrm{d}\boldsymbol{W} \in \mathbb{R}^{n_x \times n_W}$ obeys

$$\boldsymbol{U}_{k+1} = (\boldsymbol{G}_{u,k} + \boldsymbol{G}_{a,k}\boldsymbol{\Pi}_{u,k})\,\boldsymbol{U}_k + \boldsymbol{G}_{a,k}\boldsymbol{\Pi}_{W,k}, \qquad \boldsymbol{U}_0 = \boldsymbol{0}, \tag{4}$$

where $\boldsymbol{G}_{u,k} = \partial_{\boldsymbol{u}}\mathcal{G}_{\boldsymbol{\theta}}$ and $\boldsymbol{G}_{a,k} = \partial_{\boldsymbol{a}}\mathcal{G}_{\boldsymbol{\theta}}$ are the surrogate Jacobians and $\boldsymbol{\Pi}_{u,k} = \partial_{\boldsymbol{u}}\pi_{\boldsymbol{W}}$ and $\boldsymbol{\Pi}_{W,k} = \partial_{\boldsymbol{W}}\pi_{\boldsymbol{W}}$ the policy Jacobians, all evaluated at $(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k)$. The policy gradient therefore depends on the surrogate's predictions $\hat{\boldsymbol{u}}_k$ and on its Jacobians $\boldsymbol{G}_{u,k}$ and $\boldsymbol{G}_{a,k}$, which forecasting error does not measure.

Deployment. The trained policy acts on the plant as $\boldsymbol{a}_k = \pi_{\boldsymbol{W}^\star}(\boldsymbol{u}_k, \boldsymbol{r})$. Each step uses the measured state and one forward pass, with no surrogate rollout or online optimization.

Algorithm 1 Surrogate training

Require: $\mathcal{D}_{\mathrm{dyn}}$, $\mathcal{G}_{\boldsymbol{\theta}}$, horizon M
1: **repeat**
2: Sample a minibatch $\mathcal{B}$ of M-step windows
3: Roll out $\mathcal{G}_{\boldsymbol{\theta}}$ with the recorded actions
4: Compute $\mathcal{L}_{\mathrm{dyn}}(\boldsymbol{\theta})$ (Eq. 2)
5: Take a gradient step in $\boldsymbol{\theta}$
6: **until** the training criterion is met
7: **return** $\mathcal{G}_{\boldsymbol{\theta}}$

Algorithm 2 Offline policy training

Require: $\mathcal{D}_{\mathrm{DPC}}$, $\boldsymbol{r}$, frozen $\mathcal{G}_{\boldsymbol{\theta}}$, horizon N
1: **repeat**
2: Sample b initial states from $\mathcal{D}_{\mathrm{DPC}}$
3: Roll out the closed loop (Eq. 3)
4: Compute the mean of $\widehat{J}$ over the batch (Eq. 1)
5: Take a gradient step in $\boldsymbol{W}$
6: **until** the policy-training criterion is met
7: **return** $\pi_{\boldsymbol{W}^\star}$

4 What Forecast Accuracy Certifies for Control

Equation 4 shows that policy training uses both the surrogate's predictions and its Jacobians. Three claims clarify what forecast accuracy certifies about each. They adapt known facts, the simulation lemma of model-based reinforcement learning (Kearns & Singh, 2002; Luo et al., 2019; Janner et al., 2019) and the gap between uniform and derivative approximation, to the DPC setting; proofs are in Appendix G. Throughout, a fixed policy π is run from the same $\boldsymbol{u}_0$ on the plant and on the surrogate (Equation 3), with costs J and $\widehat{J}$, and $\mathcal{J}$ and $\widehat{\mathcal{J}}$ denote their expectations over $\boldsymbol{u}_0 \sim \mathcal{D}_{\mathrm{DPC}}$.

Claim 1: transfer is policy-conditioned. Let $\varepsilon_\pi(\boldsymbol{u}_0) = \max_{0\le k<N} \|\mathcal{G}_{\boldsymbol{\theta}}(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k) - \mathcal{S}(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k)\|_2$ be the largest one-step surrogate error along the surrogate's own closed loop. If the plant, the policy, and the stage cost are Lipschitz on a set containing both trajectories, then

$$\left|J(\pi; \boldsymbol{u}_0) - \widehat{J}(\pi; \boldsymbol{u}_0)\right| \le C_N\, \varepsilon_\pi(\boldsymbol{u}_0), \tag{5}$$

with C_N determined by the Lipschitz constants and the horizon N (Proposition 1). Since C_N can grow exponentially in N when the closed loop is expansive, the content of the bound is where the error is measured, not its size. Accuracy on the states and actions the policy itself generates certifies how a *fixed* policy transfers; open-loop accuracy on $\mathcal{D}_{\mathrm{dyn}}$ does not bound ε_π, and neither certifies the gradients used to obtain that policy.

Claim 2: a trained policy can fail in two ways. Write $\delta_{\mathrm{opt}} = \widehat{\mathcal{J}}(\pi_{\boldsymbol{W}^\star}) - \inf_{\pi\in\mathcal{P}} \widehat{\mathcal{J}}(\pi)$ for the optimization gap inside the surrogate and $\Delta(\pi) = |\mathcal{J}(\pi) - \widehat{\mathcal{J}}(\pi)|$ for the transfer gap. For any comparator $\pi^\circ \in \mathcal{P}$,

$$\mathcal{J}(\pi_{\boldsymbol{W}^\star}) - \mathcal{J}(\pi^\circ) \le \Delta(\pi_{\boldsymbol{W}^\star}) + \delta_{\mathrm{opt}} + \Delta(\pi^\circ). \tag{6}$$

A policy therefore underperforms either because optimization on the surrogate was poor or because its result does not transfer. Claim 1 bounds the transfer terms through trajectory-wise accuracy, while δ_{opt} is governed by the sensitivities that drive training, which Claim 3 shows accuracy does not control.

Claim 3: forward accuracy does not certify gradients. Fix an anchor action $\bar{\boldsymbol{a}}$, unit vectors $\boldsymbol{e} \in \mathbb{R}^{n_a}$ and $\boldsymbol{\phi} \in \mathbb{R}^{n_x}$, and constants $c, \eta > 0$, and let $\mathcal{G}_\eta(\boldsymbol{u}, \boldsymbol{a}) = \mathcal{S}(\boldsymbol{u}, \boldsymbol{a}) + \eta \sin(c\langle \boldsymbol{a} - \bar{\boldsymbol{a}}, \boldsymbol{e}\rangle/\eta)\, \boldsymbol{\phi}$. This surrogate is uniformly within η of the plant and its multi-step forecast error is $O(\eta)$, yet its action Jacobian differs from the plant's by c, and at parameters producing $\bar{\boldsymbol{a}}$ its one-step policy gradient differs by an amount independent of η and generically nonzero (Proposition 2). The construction requires the surrogate's curvature in the action to grow as η shrinks; when that curvature is bounded by M_2, forward accuracy does control the Jacobian error, at rate $\sqrt{2\eta M_2}$ (Appendix G). Prediction accuracy therefore certifies slope accuracy only through the surrogate's smoothness, and DPC trains on the slope.

Together the claims say that forecast error certifies neither term in Equation 6. Section 5 therefore measures forecasting and control separately and compares surrogate-side with plant-side performance, the empirical counterpart of $\Delta(\pi)$.

5 Experiments

We ask three questions, one per subsection: do forecasting and control rank surrogates consistently (Section 5.1), how do one-step and five-step FNO training compare (Section 5.2), and does a surrogate correctly report the plant performance of its own policy (Section 5.3)? Section 5.4 then tests the trained controllers under shifted initial conditions.

Table 1: PDE control environments. All use eight actuators and a spatially uniform reference r.

PDE	r	L	n_x	N	Δt	$a_{\max}$
AC	-1	2	256	100	0.01	10
Burgers	0	1	256	100	0.05	1
CH	-1	2	256	50	0.01	10
CDR	0	1	200	100	0.1	0.5
KdV	0	10	128	100	0.01	10
KS	0	22	64	200	0.1	1

Setup. We use six one-dimensional periodic PDE environments from `controlgym` (Zhang et al., 2024): Allen–Cahn (AC), viscous Burgers, Cahn–Hilliard (CH), convection–diffusion–reaction (CDR), Korteweg–de Vries (KdV), and Kuramoto–Sivashinsky (KS) (Table 1). AC and CH track the uniform state -1; the others regulate to zero. MPC rollouts on the five nonlinear problems and LQR-controlled trajectories from `controlgym` on linear CDR demonstrate regulation or tracking for the tested conditions (Appendix D.2). We compare the four surrogate configurations of Section 3, TI-DON, RNO, FNO_1, and FNO_5, with PPO trained directly on the plant using the negative stage cost as reward and the same actor architecture and actuator bounds as the DPC policy (Appendix D.1).

Metrics. All errors are endpoint mean-squared errors over the n_x grid points. For each PDE, open-loop forecasting and closed-loop tracking are evaluated at the same terminal step N listed in Table 1. The forecast error E_{pred} compares surrogate and plant from the same state under the same prescribed controls (superscript OL). The plant-side and surrogate-side tracking errors $E_{\mathrm{plant}}(\pi)$ and $E_{\mathrm{surr}}(\pi)$ measure the distance to $\boldsymbol{r}$ when the policy trained through a surrogate runs in closed loop on the plant and on that surrogate (superscript π). For evaluation rollout j,

$$\begin{gathered} E_{\mathrm{pred},j} = \tfrac{1}{n_x}\|\hat{\boldsymbol{u}}^{\mathrm{OL}}_{N,j} - \boldsymbol{u}^{\mathrm{OL}}_{N,j}\|_2^2, \\ E_{\mathrm{plant},j}(\pi) = \tfrac{1}{n_x}\|\boldsymbol{u}^{\pi}_{N,j} - \boldsymbol{r}\|_2^2, \quad E_{\mathrm{surr},j}(\pi) = \tfrac{1}{n_x}\|\hat{\boldsymbol{u}}^{\pi}_{N,j} - \boldsymbol{r}\|_2^2. \end{gathered} \tag{7}$$

Forecast errors are summarized by the median over 50 held-out rollouts. For DPC tracking, we first take the median over 50 rollouts for each trained policy, then average the $\log_{10}$ medians across three policy seeds; PPO has one policy seed per PDE. We use endpoint error rather than the cumulative cost J of Section 4 as the primary control metric because J is dominated by the initial transient: across the FNO and RNO policies it differs by less than 1% on every PDE (Figure 12), whereas the residual regulation error at the end of the horizon is where the surrogates differ.

Table 2: **Forecasting vs. plant-side tracking.** $\log_{10}$ endpoint MSE (lower is better; best per row and group in bold). Each DPC plant-side entry averages three seed-specific $\log_{10}$ medians, each over 50 rollouts; PPO uses one policy seed. Appendix J lists means, quartiles, and maxima over the evaluation rollouts.

	Forecasting				Plant-side tracking				
PDE	TI-DON	RNO	FNO_1	FNO_5	TI-DON	RNO	FNO_1	FNO_5	PPO
AC	−3.36	−3.68	−4.50	**−6.42**	−4.03	−6.71	−6.69	**−6.90**	−3.18
Burgers	−3.41	−4.71	−4.61	**−5.70**	−4.38	−5.13	−6.09	**−6.14**	−4.88
CH	−5.49	−5.38	**−5.91**	−5.66	0.02	−3.19	−5.76	**−6.39**	−4.78
CDR	−1.53	−3.11	−4.91	**−6.74**	−3.53	−7.04	−7.05	**−7.28**	−4.50
KdV	−3.54	−3.91	−3.86	**−6.36**	−3.12	−4.42	**−5.26**	−5.23	−3.69
KS	−1.75	−2.25	−3.19	**−4.34**	−2.31	−3.79	−4.40	**−5.09**	−4.15

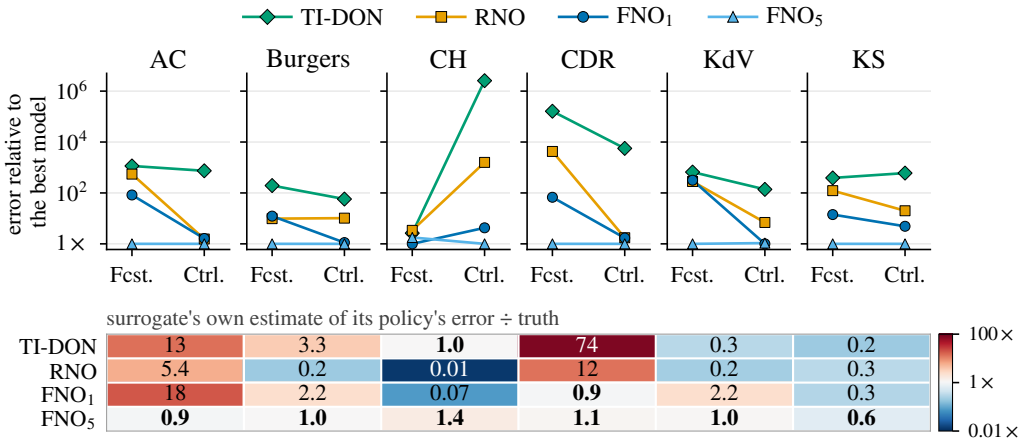

Figure 2: **Forecasting, control, and surrogate–plant agreement.** *Top:* each surrogate's error relative to the best surrogate in that PDE, for open-loop forecasting (Fcst.) and plant-side tracking (Ctrl.); 1× is the best model in each column, and crossing lines mean the two rankings disagree. *Bottom:* ρ, the error each surrogate predicts for its own policy divided by the error achieved on the plant; red is pessimistic, blue optimistic, bold agrees within a factor of two.

5.1 Forecasting and control favor different surrogate choices

Ranking the four surrogates by forecast error and by plant-side tracking error gives the same order on only two of the six benchmarks, CDR and KS (Table 2; Figure 2, top). Selecting the best forecaster remains a reasonable default: it yields the best controller on AC, Burgers, CDR, and KS. Using the seed-aggregated tracking errors in Table 2, its error is approximately four times the best on CH and only 7% higher on KdV. However, similar forecast errors can conceal large differences in controller performance, while substantially different forecast errors can yield similar tracking performance.

Similar forecasts, different controllers. On CH, the four median forecast errors differ by only a factor of 3.4 (1.24×10^{-6} to 4.19×10^{-6}), while the median tracking errors span 6.4 orders of magnitude (4.21×10^{-7} to 1.01). TI-DON has only 2.6× the forecast error of FNO_1, yet its policy leaves the plant far from the target. Figure 4 shows the resulting closed-loop behavior for one held-out initial condition: FNO_5 and PPO apply strong negative forcing early and drive the state toward -1, whereas the TI-DON policy produces small, nearly constant actions.

Different forecasts, similar controllers. On CDR, RNO has $63\times$ the median forecast error of FNO_1, yet its median tracking errors are similar (8.05×10^{-8} versus 8.59×10^{-8}). Relative to FNO_5, a $4{,}200\times$ difference in forecast error corresponds to a factor of less than twofold in tracking error. The same pattern appears on AC: RNO has $6.6\times$ the median forecast error of FNO_1, while its median tracking error is also similar.

Comparison with PPO. DPC through FNO_5 attains lower median endpoint tracking error than PPO on all six benchmarks, with margins of 0.94 to 3.72 orders of magnitude (Table 2). This comparison concerns the endpoint performance of the evaluated configurations; the two methods do not use matched plant-interaction or compute budgets (Appendix D.1).

Exploratory closed-loop diagnostics. We compare each frozen surrogate $\mathcal{G}_{\boldsymbol{\theta}}$ with the plant $\mathcal{S}_{\Delta t}$ along the closed-loop state–action pairs $(\widehat{\boldsymbol{u}}_k, \widehat{\boldsymbol{a}}_k)$ generated by the policy trained through that surrogate. We measure the one-step on-policy defect $d_k = \|\mathcal{G}_{\boldsymbol{\theta}} - \mathcal{S}_{\Delta t}\|_2$, its trajectory maximum $\varepsilon_\pi = \max_k d_k$, the relative action-Jacobian mismatch $E_a(k)$, and the relative state-Jacobian mismatch $E_u^{\text{probe}}(k)$ over fixed probe directions, all evaluated at $(\widehat{\boldsymbol{u}}_k, \widehat{\boldsymbol{a}}_k)$ (Appendix I).

Figure 3 reports the on-policy diagnostics for one checkpoint per surrogate–policy pair across 16 held-out CH initial conditions. RNO has the largest trajectory-level defect, yet its plant-side tracking error remains far below TI-DON's and above those of the FNO controllers. The defect maximum therefore does not rank terminal control quality. The state-sensitivity diagnostic separates TI-DON more clearly: its E_u^{probe} is substantially larger than for RNO or either FNO. The action-Jacobian mismatch E_a varies less across models, with RNO giving the lowest values over most of the trajectory. This suggests a state-sensitivity problem for TI-DON, but does not show that it causes the poor control. Its policy also fails to track inside the surrogate.

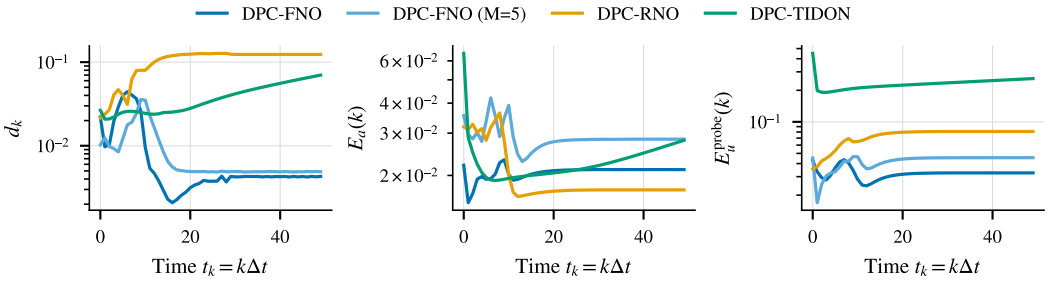

Figure 3: Closed-loop sensitivity diagnostics along the surrogate rollout for Cahn–Hilliard (CH) $t_k = k\Delta t$ with $N\Delta t = T$. Each curve is the median over 16 held-out initial conditions.

5.2 One-step versus five-step FNO training

FNO_5 achieves the lowest median tracking error on five of the six benchmarks and is within 7% of the lowest on KdV, but improvements in forecasting and control do not consistently align. On KdV, FNO_5 improves median forecast error by more than two orders of magnitude while slightly worsening tracking. On CH, the pattern reverses: FNO_5 has higher median forecast error but reduces median tracking error approximately fourfold. The CH forecast ranking also depends on the summary statistic, since FNO_5 has the lower mean error. Because the two configurations also differ in learning-rate schedule, training duration, and training windows (Appendix C.1), this comparison characterizes the full training configurations rather than isolating rollout depth M.

The selected-checkpoint diagnostics likewise show that lower local error does not consistently yield lower endpoint error (Figure 10). Relative to FNO_1, FNO_5 reduces the median trajectory-maximum defect and action-Jacobian mismatch by factors of 9.6 and 4.2 on CDR, and 5.1 and 7.8 on KdV. The probed state-Jacobian mismatch decreases by factors of 5.4 and 1.2, respectively. On the same diagnostic initial conditions, FNO_5 has $4.39\times$ lower median endpoint error on CDR but 7.9% higher error on KdV. On CH, FNO_1 has lower probed state-Jacobian mismatch (0.032 versus 0.042), while FNO_5 has $1.96\times$ lower endpoint error (Figure 3). These exploratory comparisons use one checkpoint per surrogate–policy pair and follow each policy along its own closed-loop trajectory, with fewer initial conditions than the main 50-rollout, three-policy-seed evaluation. They therefore do not provide an architecture-level ranking of intrinsic Jacobian accuracy.

5.3 Do surrogates report their own controller's performance?

Section 4 shows that transfer depends on surrogate accuracy along the policy's own trajectories. We summarize whether a surrogate predicts its own policy's terminal tracking error by the ratio of medians

$$\rho(\pi) = \text{median}_j\, E_{\text{surr},j}(\pi) \,/\, \text{median}_j\, E_{\text{plant},j}(\pi), \tag{8}$$

where $\rho(\pi) < 1$ means the surrogate is optimistic and $\rho(\pi) > 1$ means it is pessimistic. The largest mismatches are about 74-fold pessimism for TI-DON on CDR and 85-fold optimism ($\rho = 0.012$) for RNO on CH (Table 9). FNO_5 is the most consistent configuration, with ratios from 0.60 to 1.45, so it stays within a factor of two on all six PDEs. Agreement does not establish controller quality: on CH, the reported TI-DON ratio is approximately one while both endpoint errors are of order 1, that is, no tracking at all. The ratio summarizes terminal-error medians, not individual trajectories or the cumulative-cost gap of Equation 5. Appendix H defines the aggregation across policy seeds.

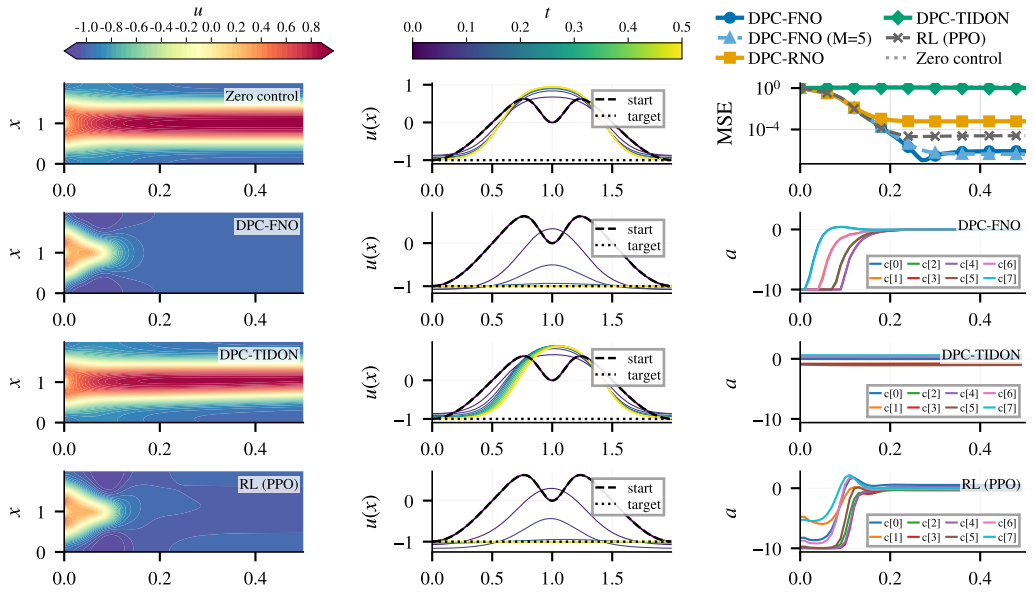

Figure 4: **Closed-loop behavior on Cahn–Hilliard** for one held-out initial condition and target -1. *Rows, top to bottom:* zero control, DPC through FNO_5, DPC through TI-DON, and PPO. *Left:* state evolution. *Center:* spatial profiles at selected times. *Right:* tracking MSE over time and the eight actuator amplitudes. Despite comparable forecast accuracy, the TI-DON policy leaves the state far from the target.

5.4 Shifted initial conditions

We perturb the held-out initial states in three ways, keeping comparisons paired across shift levels (Appendix E.1): amplitude scaling by α, additive Gaussian perturbations of relative level χ, and burn-in perturbations, in which each state evolves for 10 uncontrolled steps under process noise of level χ. AC, CH, and KS remain comparatively stable across all levels, Burgers is sensitive to amplitude scaling to a degree that depends on the surrogate, and KdV degrades sharply as the disturbance grows (Figure 5). CH policies fail from some initial states at the largest perturbations; a plausible but untested cause is that the surrogate, trained on noise-free trajectories, is queried on perturbed closed-loop states.

The shifted-state diagnostics reveal two distinct behaviors (Figure 11). First, surrogate-rollout stability need not match plant-policy reliability under distribution shift. On CH at burn-in level $\chi = 0.2$, the FNO_5 surrogate rollout terminates for all 16 initial conditions within steps 3–5, while none of the corresponding plant rollouts terminate and the median plant endpoint MSE remains 4.18×10^{-7}. Second, the time evolution of the defect can matter beyond its trajectory maximum ε_π. On KdV at

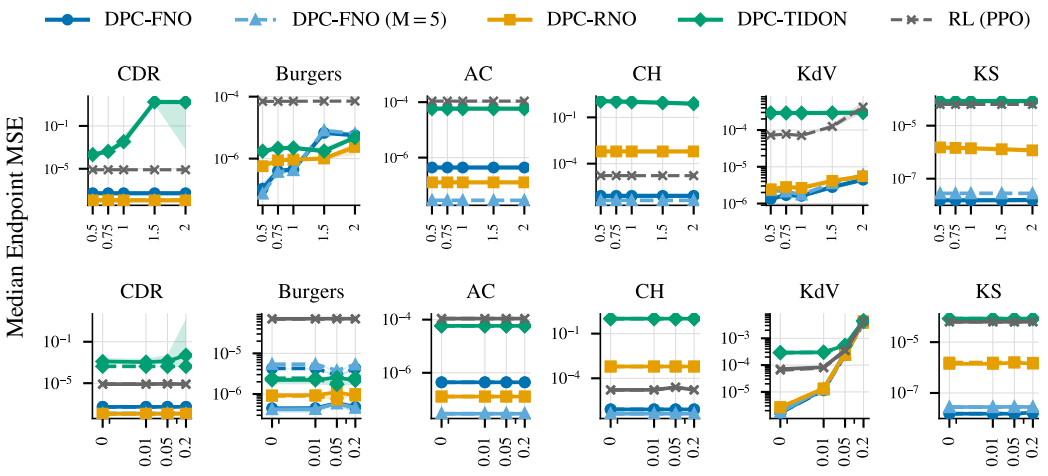

Figure 5: Out-of-distribution robustness of DPC surrogates for the PDE environments. Both rows show median plant-side endpoint MSE with shaded interquartile-range bands over $n = 20$ evaluation initial conditions per point. **Top row:** Endpoint MSE versus initial-condition amplitude scale α. **Bottom row:** Endpoint MSE versus disturbance level χ for additive perturbations (solid lines) and burn-in perturbations (dashed lines).

$\chi = 0.2$, the median defect d_k falls from 0.762 at $k = 0$ to 0.0318 at $k = 5$ and 0.0092 at $k = 99$, even though the plant endpoint MSE rises to approximately 3.45×10^{-3}.

6 Discussion and Conclusion

Across six PDE benchmarks, open-loop forecast error did not reliably rank the controllers trained through neural-operator surrogates. On Cahn–Hilliard, similar median forecast errors accompanied more than six orders of magnitude of variation in median terminal tracking MSE. In another comparison, a 63-fold forecast-error difference corresponded to tracking errors within 6%. We therefore report forecasting under prescribed controls, tracking by the policy inside its surrogate, and tracking by that policy on the numerical plant. These evaluations measure prediction accuracy, surrogate-side control, and plant-side control, respectively. The ratio $\rho(\pi)$ identifies surrogates that misreport the performance of the controllers trained through them.

FNO_5 was the most reliable configuration among those tested by these evaluations. It achieved the lowest median terminal tracking error on five of six benchmarks and came within 7% of the lowest on the sixth. It also gave the most consistent self-reports among the tested configurations: its surrogate-side and plant-side tracking errors agreed within a factor of two on all six benchmarks. Its median terminal tracking error was 0.94 to 3.72 orders of magnitude lower than PPO's. Agreement alone does not imply good plant-side control. FNO_5 and FNO_1 also differ in optimization settings, so these results do not isolate the effect of supervision horizon.

The analysis in Section 4 is consistent with, but does not explain, these results: the transfer bound depends on errors along trajectories induced by the policy, while the gradient counterexample shows that small forward errors need not yield accurate policy gradients. Selected-checkpoint diagnostics expose transition and sensitivity errors missed by open-loop forecasting, but their magnitudes do not consistently rank controller performance. Under strongly perturbed Cahn–Hilliard initial conditions, an FNO_5 surrogate rollout fails while its policy still tracks on the plant. These observations distinguish failure modes; they do not establish a causal role for Jacobian error or a general reliability threshold.

Limitations. The benchmarks are one-dimensional periodic PDEs with low-dimensional initial-condition families, fixed actuator layouts, and constant references. Every DPC configuration uses three independently trained policy seeds, but each surrogate is trained once, so surrogate-training variability is not characterized. Endpoint MSE omits control effort and transients, and the pooled rollout statistics combine policy-seed and initial-condition variation rather than isolating either source. The PPO comparison is not budget-matched.

AI USE STATEMENT

Generative AI tools assisted with manuscript drafting and editing, LaTeX preparation, literature identification, numerical transcription and arithmetic checks. Numerical comparisons were checked against the original experimental tables. These checks do not constitute independent reruns of the experiments. The authors take responsibility for the final manuscript, including the accuracy of its results, citations, mathematical claims, and AI-assisted content.

REPRODUCIBILITY STATEMENT

Appendix B gives the general parameterized control and DPC formulation, Appendix C specifies the surrogate and policy architectures and hyperparameters, Appendix E the data generation and shifted initial conditions, Appendix F the environments, and Appendix G the proofs of Propositions 1 and 2. The code and supporting materials used to reproduce the experiments are publicly available on Zenodo at `doi:10.5281/zenodo.22970373`.

REFERENCES

Brandon Amos, Ivan Jimenez, Jacob Sacks, Byron Boots, and J. Zico Kolter. Differentiable MPC for end-to-end planning and control. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Joel A. E. Andersson, Joris Gillis, Greg Horn, James B. Rawlings, and Moritz Diehl. CasADi – a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019. doi: 10.1007/s12532-018-0139-4.

Jannis Becktepe, Aleksandra Franz, Nils Thuerey, and Sebastian Peitz. Plug-and-play benchmarking of reinforcement learning algorithms for large-scale flow control. *arXiv preprint arXiv:2601.15015*, 2026.

Peter Benner, Serkan Gugercin, and Karen Willcox. A survey of projection-based model reduction methods for parametric dynamical systems. *SIAM Review*, 57(4):483–531, 2015.

Luke Bhan, Yuanyuan Shi, and Miroslav Krstic. Neural operators for hyperbolic PDE backstepping feedback laws. In *IEEE Conference on Decision and Control (CDC)*, pp. 5994–5999, 2023.

Luke Bhan, Yuanyuan Shi, and Miroslav Krstic. Neural operators for bypassing gain and control computations in PDE backstepping. *IEEE Transactions on Automatic Control*, 69:5310–5325, 2024.

Steven L. Brunton and Bernd R. Noack. Closed-loop turbulence control: Progress and challenges. *Applied Mechanics Reviews*, 67(5):050801, 2015.

Michele Alessandro Bucci, Onofrio Semeraro, Alexandre Allauzen, Guillaume Wisniewski, Laurent Cordier, and Lionel Mathelin. Control of chaotic systems by deep reinforcement learning. *Proceedings of the Royal Society A*, 475(2231):20190351, 2019.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Steven M. Cox and Paul C. Matthews. Exponential time differencing for stiff systems. *Journal of Computational Physics*, 176(2):430–455, 2002.

Wojciech M. Czarnecki, Simon Osindero, Max Jaderberg, Grzegorz Świrszcz, and Razvan Pascanu. Sobolev training for neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Thomas O. de Jong, Khemraj Shukla, and Mircea Lazar. Deep operator neural network model predictive control. *arXiv preprint arXiv:2505.18008*, 2025.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning (ICML)*, pp. 465–472, 2011.

Ján Drgoňa, Karol Kiš, Aaron Tuor, Draguna Vrabie, and Martin Klaučo. Differentiable predictive control: Deep learning alternative to explicit model predictive control for unknown nonlinear systems. *Journal of Process Control*, 116:80–92, 2022.

Ján Drgoňa, Aaron Tuor, and Draguna Vrabie. Learning constrained parametric differentiable predictive control policies with guarantees. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 54(6):3596–3607, 2024.

Lawrence C. Evans. *Partial Differential Equations*, volume 19 of *Graduate Studies in Mathematics*. American Mathematical Society, 2 edition, 2010.

Amir-massoud Farahmand, André Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

Somdatta Goswami, Katiana Kontolati, Michael D. Shields, and George Em Karniadakis. Deep transfer operator learning for partial differential equations under conditional shift. *Nature Machine Intelligence*, 4:1155–1164, 2022.

Somdatta Goswami, Aniruddha Bora, Yue Yu, and George Em Karniadakis. Physics-informed deep neural operator networks. In *Machine Learning in Modeling and Simulation: Methods and Applications*, Computational Methods in Engineering & the Sciences, pp. 219–254. Springer, 2023.

Max D. Gunzburger. *Perspectives in Flow Control and Optimization*. SIAM, 2003.

Yusuf Güven, Vincenzo Di Vito, and Ferdinando Fioretto. Learning to solve optimization problems constrained with partial differential equations. *arXiv preprint arXiv:2509.24573*, 2025.

Michael Hinze, René Pinnau, Michael Ulbrich, and Stefan Ulbrich. *Optimization with PDE Constraints*, volume 23 of *Mathematical Modelling: Theory and Applications*. Springer, 2009.

Philipp Holl and Nils Thuerey. ϕ_{flow} (PhiFlow): Differentiable simulations for PyTorch, TensorFlow and Jax. In *International Conference on Machine Learning (ICML)*, 2024.

Philipp Holl, Vladlen Koltun, and Nils Thuerey. Learning to control PDEs with differentiable physics. In *International Conference on Learning Representations (ICLR)*, 2020.

Rakhoon Hwang, Jae Yong Lee, Jin Young Shin, and Hyung Ju Hwang. Solving PDE-constrained control problems using operator learning. In *AAAI Conference on Artificial Intelligence*, volume 36, pp. 4504–4512, 2022.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Aly-Khan Kassam and Lloyd N. Trefethen. Fourth-order time-stepping for stiff PDEs. *SIAM Journal on Scientific Computing*, 26(4):1214–1233, 2005.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2):209–232, 2002.

John Kim and Thomas R. Bewley. A linear systems approach to flow control. *Annual Review of Fluid Mechanics*, 39:383–417, 2007.

Katiana Kontolati, Somdatta Goswami, George Em Karniadakis, and Michael D. Shields. Learning nonlinear operators in latent spaces for real-time predictions of complex dynamics in physical systems. *Nature Communications*, 15:5101, 2024.

Basil Kouvaritakis and Mark Cannon. *Model Predictive Control: Classical, Robust and Stochastic*. Springer, 2016.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to PDEs. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Miroslav Krstic and Andrey Smyshlyaev. *Boundary Control of PDEs: A Course on Backstepping Designs*. SIAM, 2008.

Miroslav Krstic, Luke Bhan, and Yuanyuan Shi. Neural operators of backstepping controller and observer gain functions for reaction–diffusion PDEs. *Automatica*, 164:111649, 2024.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. In *Learning for Dynamics and Control Conference (L4DC)*, 2020.

Randall J. LeVeque. *Finite Difference Methods for Ordinary and Partial Differential Equations: Steady-State and Time-Dependent Problems*. SIAM, 2007.

Peilun Li, Kaiyuan Tan, and Thomas Beckers. NAPI-MPC: Neural accelerated physics-informed MPC for nonlinear PDE systems. Technical report, Technical report, 2025.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Phillip Lippe, Bastiaan S. Veeling, Paris Perdikaris, Richard E. Turner, and Johannes Brandstetter. PDE-Refiner: Achieving accurate long rollouts with neural PDE solvers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Miguel Liu-Schiaffini, Clare E Singer, Nikola Kovachki, Sze Chai Leung, Hyunji Jane Bae, Kamyar Azizzadenesheli, and Anima Anandkumar. Tipping point forecasting in non-stationary dynamics on function spaces. *arXiv preprint arXiv:2308.08794*, 2023.

Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

Oliver Lundqvist and Fabricio Oliveira. Employing deep neural operators for PDE control by decoupling training and optimization. *arXiv preprint arXiv:2506.04742*, 2025.

Yuping Luo, Huazhe Xu, Yuanzhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations*, 2019.

Fabio Muratore, Fabio Ramos, Greg Turk, Wenhao Yu, Michael Gienger, and Jan Peters. Robot learning from randomized simulations: A review. *Frontiers in Robotics and AI*, 9:799893, 2022.

Dibyajyoti Nayak and Somdatta Goswami. TI-DeepONet: Learnable time integration for stable long-term extrapolation. *arXiv preprint arXiv:2505.17341*, 2025.

Yangchen Pan, Amir-massoud Farahmand, Martha White, Saleh Nabi, Piyush Grover, and Daniel Nikovski. Reinforcement learning with function-valued action spaces for partial differential equation control. In *International Conference on Machine Learning (ICML)*, 2018.

Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3803–3810, 2018.

Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, 2000.

Jie Qi, Jiaqi Hu, Jing Zhang, and Miroslav Krstic. Neural operator feedback for a first-order PIDE with spatially-varying state delay. *arXiv preprint arXiv:2412.08219*, 2024.

Jean Rabault, Miroslav Kuchta, Atle Jensen, Ulysse Réglade, and Nicolas Cerardi. Artificial neural networks trained through deep reinforcement learning discover control strategies for active flow control. *Journal of Fluid Mechanics*, 865:281–302, 2019.

Dibakar Roy Sarkar, Ján Drgoňa, and Somdatta Goswami. Learning to control PDEs with differentiable predictive control and time-integrated neural operators. *arXiv preprint arXiv:2511.08992*, 2025.

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

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Dan MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. PDEBench: An extensive benchmark for scientific machine learning. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2022.

Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 23–30, 2017.

Lloyd N. Trefethen. *Finite Difference and Spectral Methods for Ordinary and Partial Differential Equations*. Cornell University, 1996.

Lloyd N. Trefethen. *Spectral Methods in MATLAB*. SIAM, 2000.

Fredi Tröltzsch. *Optimal Control of Partial Differential Equations: Theory, Methods and Applications*, volume 112 of *Graduate Studies in Mathematics*. American Mathematical Society, 2010.

Kiwon Um, Robert Brand, Yun (Raymond) Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Stefan Werner and Sebastian Peitz. Learning a model is paramount for sample efficiency in reinforcement learning control of pdes. *arXiv preprint arXiv:2302.07160*, 2023.

Pietro Zanotta, Dibakar Roy Sarkar, Honghui Zheng, Somdatta Goswami, and Ján Drgoňa. CINOC: Cardinality-invariant neural operator policies for scalable PDE control. *arXiv preprint arXiv:2605.25867*, 2026.

Kevin Zeng, Alec J. Linot, and Michael D. Graham. Data-driven control of spatiotemporal chaos with reduced-order neural ODE-based models and reinforcement learning. *Proceedings of the Royal Society A*, 478(2267):20220297, 2022.

Xiangyuan Zhang, Weichao Mao, Saviz Mowlavi, Mouhacine Benosman, and Tamer Başar. Controlgym: Large-scale control environments for benchmarking reinforcement learning algorithms. In *Learning for Dynamics and Control Conference (L4DC)*, pp. 181–196, 2024.

Zelin Zhao, Zongyi Li, Kimia Hassibi, Kamyar Azizzadenesheli, Junchi Yan, H Jane Bae, Di Zhou, and Anima Anandkumar. Physics-informed neural-operator predictive control for drag reduction in turbulent flows. *arXiv preprint arXiv:2510.03360*, 2025.

A NOTATION

Table 3: Notation used in Sections 3 and 4.

Symbol	Meaning
$\Omega = [0, L]$, n_x	Periodic spatial domain and number of grid points
Δt, N, k	Control interval, control horizon (number of steps), step index
$\mathcal{F}$	Uncontrolled spatial differential operator of the PDE
Φ_q, n_a	Spatial profile of actuator q; number of actuators
$\boldsymbol{a}_k \in \mathbb{A}$, $a_{\max}$	Actuator amplitudes at step k; $\mathbb{A} = \{\boldsymbol{a} : \|\boldsymbol{a}\|_\infty \le a_{\max}\}$
$\boldsymbol{u}_k$, $\boldsymbol{r}$	Discretized plant state and fixed reference, both in $\mathbb{R}^{n_x}$
$\mathcal{S}$	Plant: one control interval of the numerical solver
$\mathcal{G}_{\boldsymbol{\theta}}$, $\boldsymbol{\theta}$	Surrogate one-step map and its parameters
$\hat{\boldsymbol{u}}_k$, $\hat{\boldsymbol{a}}_k$	Surrogate-predicted state and the corresponding policy action
$\pi_{\boldsymbol{W}}$, $\boldsymbol{W}$, n_W	Feedback policy, its parameters, and their number
$\boldsymbol{W}^\star$	Parameters returned by policy training
ℓ, $\boldsymbol{Q}$, $\boldsymbol{R}$	Stage cost and its state and action weights
$\mathcal{L}_{\text{train}}$, λ_{aux}, b	Policy-training loss, auxiliary weight, and minibatch size
J, $\widehat{J}$	Finite-horizon cost evaluated on the plant and on the surrogate
$\mathcal{D}_{\text{dyn}}$	Open-loop trajectories used to train the surrogate
$\mathcal{D}_{\text{DPC}}$	Initial-state distribution used for policy training
M, $\mathcal{B}$	Surrogate supervision horizon; minibatch of training windows
$\boldsymbol{U}_k$	Total sensitivity $\mathrm{d}\hat{\boldsymbol{u}}_k/\mathrm{d}\boldsymbol{W}$
$\boldsymbol{G}_{u,k}$, $\boldsymbol{G}_{a,k}$	Surrogate Jacobians with respect to state and action
$\boldsymbol{\Pi}_{u,k}$, $\boldsymbol{\Pi}_{W,k}$	Policy Jacobians with respect to state and parameters
ε_π	Largest one-step surrogate error along the surrogate closed loop
L_u, L_a, L_π, L_ℓ	Lipschitz constants of plant (state, action), policy, and stage cost
γ, Σ_N, C_N	Closed-loop error growth factor, its horizon-averaged sum, and $C_N = L_\ell(1 + L_\pi)\Sigma_N$
$\bar{\boldsymbol{W}}$, $\bar{\boldsymbol{a}}$, $\bar{\boldsymbol{B}}$	Anchor parameters, anchor action, and policy parameter Jacobian (Prop. 2)
c, η, $\boldsymbol{e}$, $\boldsymbol{\phi}$	Jacobian-error size, perturbation amplitude, action and state directions (Prop. 2)
$\mathcal{P}$, δ_{opt}, $\Delta(\pi)$	Policy class, surrogate optimization gap, and transfer gap (App. G.2)

B GENERAL PDE-CONTROL AND DPC FORMULATION

Parameterized PDE-constrained control. Let $u : \Omega \times [0, T] \to \mathbb{R}$ be a scalar field on the periodic domain $\Omega = [0, L]^d \subset \mathbb{R}^d$. A general controlled instance satisfies

$$\partial_t u(\boldsymbol{x}, t) = \mathcal{F}(u, \nabla_{\boldsymbol{x}} u, \ldots, \nabla^p_{\boldsymbol{x}} u; \boldsymbol{\xi}(t))\,(\boldsymbol{x}, t) + f(\boldsymbol{x}, t), \quad (\boldsymbol{x}, t) \in \Omega \times (0, T], \tag{9a}$$

$$u(\boldsymbol{x}, 0) = u_0(\boldsymbol{x}), \tag{9b}$$

$$\partial^\nu_{x_i} u\big|_{x_i=0} = \partial^\nu_{x_i} u\big|_{x_i=L}, \qquad \nu = 0, \ldots, p-1, \quad i = 1, \ldots, d, \tag{9c}$$

where $\boldsymbol{\xi}(t) \in \mathbb{R}^{n_\xi}$ collects time-varying physical parameters or constraint bounds. For actuator profiles $\Phi_q : \Omega \to \mathbb{R}$ and amplitudes $\boldsymbol{a}(t) \in \mathbb{A}$, the distributed forcing is

$$f(\boldsymbol{x}, t) = \sum_{q=1}^{n_a} \Phi_q(\boldsymbol{x}) a_q(t), \qquad \mathbb{A} = \{\boldsymbol{a} \in \mathbb{R}^{n_a} : \|\boldsymbol{a}\|_\infty \le a_{\max}\}. \tag{10}$$

An instance is specified by $(u_0, r, \boldsymbol{\xi})$, where $r : \Omega \to \mathbb{R}$ is the reference field. Its continuous-time control problem is

$$\begin{aligned} \min_{\boldsymbol{a}(\cdot)} \quad & J_{\text{cont}}[u, \boldsymbol{a}] = \int_0^T \ell(u(\cdot, t), \boldsymbol{a}(t); r, \boldsymbol{\xi}(t))\, dt + \ell_T(u(\cdot, T); r), \\ \text{s.t.} \quad & h(u(\cdot, t), \boldsymbol{\xi}(t)) \le 0, \qquad g(\boldsymbol{a}(t), \boldsymbol{\xi}(t)) \le 0, \end{aligned} \tag{11}$$

subject to Equation 9 and the forcing in Equation 10. Here, ℓ and ℓ_T are running and terminal cost functionals, while h and g encode state and actuator constraints; vector inequalities are interpreted componentwise.

Parametric problem. Solving Equation 11 separately for each instance produces an instance-specific open-loop input. The parametric problem instead learns a stationary feedback policy $\pi_{\boldsymbol{W}}$ across an instance distribution $\mathcal{D}_{\text{inst}}$:

$$\min_{\boldsymbol{W}\in\mathbb{R}^{n_W}} \mathbb{E}_{(u_0,r,\boldsymbol{\xi})\sim\mathcal{D}_{\text{inst}}} \left[J_{\text{cont}}[u,\boldsymbol{a}]\right], \qquad \boldsymbol{a}(t) = \pi_{\boldsymbol{W}}(u(\cdot,t), r, \boldsymbol{\xi}(t)). \tag{12}$$

The policy parameters $\boldsymbol{W}$ concatenate all trainable weights and biases. The expectation accounts for variation in initial conditions, references, and problem parameters.

Discrete DPC formulation. Let $\{\boldsymbol{x}_i\}_{i=1}^{n_x}$ be a spatial grid, let $t_k = k\Delta t$ with $N\Delta t = T$, and let $\boldsymbol{u}_k, \boldsymbol{r} \in \mathbb{R}^{n_x}$ and $\boldsymbol{\xi}_k = \boldsymbol{\xi}(t_k) \in \mathbb{R}^{n_\xi}$ denote the discretized state, reference, and parameter vector. The actuator matrix $\boldsymbol{\Phi} \in \mathbb{R}^{n_x\times n_a}$ has entries $\Phi_{i,q} = \Phi_q(\boldsymbol{x}_i)$, so $\boldsymbol{f}_k = \boldsymbol{\Phi}\boldsymbol{a}_k$. The surrogate and policy maps are $\mathcal{G}_{\boldsymbol{\theta}} : \mathbb{R}^{n_x}\times\mathbb{A}\times\mathbb{R}^{n_\xi} \to \mathbb{R}^{n_x}$ and $\pi_{\boldsymbol{W}} : \mathbb{R}^{n_x}\times\mathbb{R}^{n_x}\times\mathbb{R}^{n_\xi} \to \mathbb{A}$, respectively. Given fixed surrogate parameters $\boldsymbol{\theta}$, DPC approximates Equation 12 by

$$\min_{\boldsymbol{W}} \quad \mathbb{E}_{(\boldsymbol{u}_0,\boldsymbol{r},\boldsymbol{\xi})\sim\mathcal{D}_{\text{inst}}} \left[\frac{1}{N}\sum_{k=0}^{N-1} \ell(\boldsymbol{u}_{k+1}, \boldsymbol{a}_k; \boldsymbol{r}, \boldsymbol{\xi}_k) + \ell_T(\boldsymbol{u}_N; \boldsymbol{r})\right], \tag{13a}$$

$$\text{s.t.} \quad \boldsymbol{u}_{k+1} = \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u}_k, \boldsymbol{a}_k; \boldsymbol{\xi}_k), \qquad \boldsymbol{a}_k = \pi_{\boldsymbol{W}}(\boldsymbol{u}_k, \boldsymbol{r}, \boldsymbol{\xi}_k), \tag{13b}$$

$$h(\boldsymbol{u}_k, \boldsymbol{\xi}_k) \le \boldsymbol{0}, \qquad g(\boldsymbol{a}_k, \boldsymbol{\xi}_k) \le \boldsymbol{0}, \qquad k = 0, \dots, N-1. \tag{13c}$$

Automatic differentiation through this closed-loop rollout gives gradients of the objective with respect to $\boldsymbol{W}$ while $\boldsymbol{\theta}$ remains fixed. Section 3 specializes this formulation to $d = 1$, fixed PDE parameters, the bounded policy codomain $\mathbb{A}$, and the quadratic stage cost used in Equation 1; the fixed reference is suppressed from the arguments of ℓ. The experiments do not use separate terminal or soft state-constraint penalties.

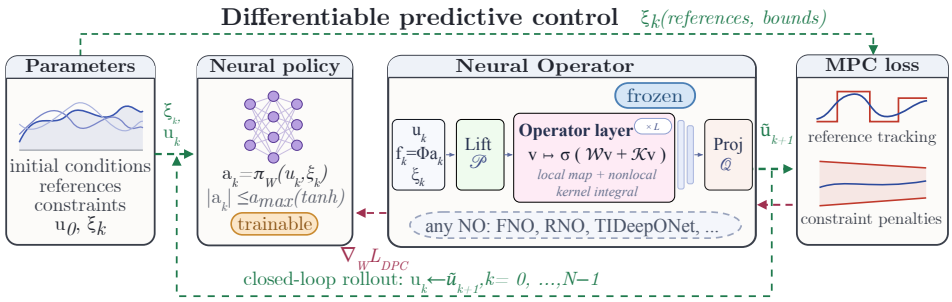

Figure 6: **Neural-operator surrogates for differentiable predictive control.** A surrogate is fitted to open-loop trajectories and frozen. The feedback policy is then optimized through surrogate closed-loop rollouts and deployed on the numerical plant.

C Architectures and Training Details

C.1 Surrogate models

Surrogate architectures. All surrogates define a one-step map $\mathcal{G}_{\boldsymbol{\theta}} : \mathbb{R}^{n_x}\times\mathbb{A} \to \mathbb{R}^{n_x}$ on the physical state. FNO predicts the next state with a Fourier neural operator backbone (Li et al., 2021; Kovachki et al., 2023). RNO uses recurrent spectral blocks (Liu-Schiaffini et al., 2023); the selected configurations reset the hidden state before every call, so RNO also acts as a one-step map. TI-DeepONet (TI-DON) learns the temporal derivative of the state and integrates it with a fourth-order Runge–Kutta step (Nayak & Goswami, 2025). State representations, control injection, and hyperparameters are given below. Surrogate training budgets were matched approximately in wall-clock time ($\approx$ 12 per configuration on A100), so epoch counts differ across architectures.

Surrogate inputs. FNO and RNO receive the physical state together with a spatial representation of the actuator amplitudes. The actuator vector is lifted through $\boldsymbol{a}_{k,\text{lift}} = \boldsymbol{B}_{\text{lift}}\boldsymbol{a}_k$, where $\boldsymbol{B}_{\text{lift}} \in \mathbb{R}^{n_x\times n_a}$, and concatenated with the state as an input channel. This representation is distinct from the physical forcing in Equation 9. TI-DON instead receives the actuator vector through a separate control branch.

Table 4: RNO architecture settings of the selected checkpoints.

PDE	Modes	Width	Layers
AC	32	32	3
Burgers	28	32	3
CH	32	32	3
CDR	24	32	3
KdV	32	32	3
KS	32	32	3

Table 5: Reported TI-DON architecture settings. Bracketed lists give hidden-layer widths.

PDE	State branch	Control branch	Trunk	n_b
AC	$[192, 192, 192]$	$[32, 32, 32]$	$[64, 64, 64]$	192
Burgers	$[192, 192, 192]$	$[32, 32, 32]$	$[64, 64, 64]$	192
CH	$[256, 256, 256]$	$[64, 64, 64]$	$[128, 128, 128]$	256
CDR	$[128, 128, 128]$	$[32, 32, 32]$	$[64, 64, 64]$	128
KdV	$[192, 192, 192]$	$[32, 32, 32]$	$[128, 128, 128]$	192
KS	$[256, 256, 256]$	$[64, 64, 64]$	$[128, 128, 128]$	256

Checkpoint-selection splits. Each surrogate configuration loads the same $2{,}500$ open-loop trajectories and uses seed 0. FNO and RNO train on the first $2{,}000$ complete trajectories and reserve the remaining 500 for checkpoint selection. TI-DON first flattens all one-step transitions across trajectories, shuffles them with seed 0, and then applies an 80/20 split. Its training and checkpoint-selection sets can therefore contain transitions from the same trajectory. These internal splits are distinct from the 50 rollout evaluation set used in the result tables.

FNO. Both FNO configurations use 32 Fourier modes, width 64, four layers, no domain padding, batch size 128, Adam, and weight decay 10^{-4}. FNO_1 uses learning rate 10^{-3} and trains for $2{,}500$ epochs, except on CDR, where it trains for $5{,}000$. A StepLR schedule halves the learning rate every 625 epochs, or every $1{,}250$ epochs on CDR. The selected checkpoint minimizes one-step relative L^2 error on the reserved trajectories.

FNO_5 is initialized independently rather than fine-tuned from FNO_1. It trains for $1{,}000$ epochs on stride-one five-step windows using learning rate 5×10^{-4}. Predicted states are fed back autoregressively, the controls remain the recorded controls, and the five relative L^2 terms have equal weight. Its StepLR schedule halves the learning rate every $250{,}000$ minibatch updates. The selected checkpoint minimizes the mean validation loss across the five prediction depths.

Scope of the FNO_1 versus FNO_5 comparison. FNO_1 and FNO_5 share the architecture, data, batch size, seed, optimizer family, and weight decay, but also differ in learning-rate schedule, number of epochs, and construction of training windows. Their comparison is therefore at the level of training configurations rather than a controlled ablation of the supervision horizon M.

RNO. The reported RNO has three recurrent spectral layers, width 32, no domain padding, and the environment-specific Fourier mode counts in Table 4. Each selected configuration sets both inference warm-up and hidden-state propagation to zero. The implementation therefore reinitializes its hidden state before every surrogate evaluation and realizes the physical-state one-step map used in Section 3.

RNO trains for 600 epochs with batch size 128, Adam, learning rate 10^{-3}, and weight decay 10^{-4}. StepLR halves the learning rate every 150 epochs, except on CDR, where the interval is 100 epochs. The selected checkpoint minimizes held-out one-step relative L^2 error.

TI-DON. TI-DON learns a temporal-derivative operator and integrates it using classical fourth-order Runge–Kutta. Its state branch, control branch, and spatial trunk have the hidden-layer widths in Table 5. The latent dimension is denoted by n_b.

TI-DON uses batch size 100, Adam with initial learning rate 10^{-3}, and the schedule $\alpha_s = 10^{-3}(0.9)^{s/2000}$ at optimizer step s. The configurations specify 100,000 total update steps. Every 100 steps, checkpoint selection compares the current model on one randomly sampled batch of 100 transitions from the 20% split rather than on the entire split.

C.2 Policy training

DPC policy. The policy concatenates the current state, reference, and any supplied PDE parameters, then applies three hidden layers of width 128 with SiLU activations. A final componentwise $a_{\max}\tanh(\cdot)$ transformation maps its output to $\mathbb{A}$. The experimental objective uses $\boldsymbol{Q} = 0.1\boldsymbol{I}_{n_x}$ and $\boldsymbol{R} = 10^{-3}\boldsymbol{I}_{n_a}$ in the stage cost of Section 3. Each run loads 500 initial states from $\mathcal{D}_{\mathrm{DPC}}$ and uses a seeded random 80/20 split. Policy training uses the full control horizon N, batch size 64 except on CDR where it is 50, Adam with learning rate 10^{-3}, and the schedule $\alpha_s = 10^{-3}(0.9)^{s/2000}$ at update s. Gradients are clipped to norm 1, and training ends after 10,000 updates. Every 100 updates, the checkpoint metric is evaluated on a fixed set of eight initial conditions from the held-out split; the selected policy has the lowest such full-horizon surrogate loss.

Training objective. For a minibatch $\{\boldsymbol{u}_{0,j}\}_{j=1}^{b}$ drawn from $\mathcal{D}_{\mathrm{DPC}}$, the reported runs minimize

$$\mathcal{L}_{\mathrm{train}}(\boldsymbol{W}) = \frac{1}{b}\sum_{j=1}^{b}\Big[\widehat{J}(\pi_{\boldsymbol{W}};\boldsymbol{u}_{0,j}) + \lambda_{\mathrm{aux}}\,(\hat{\boldsymbol{u}}_{N-25,j}-\boldsymbol{r})^{\top}\boldsymbol{Q}\,(\hat{\boldsymbol{u}}_{N-25,j}-\boldsymbol{r})\Big], \tag{14}$$

where $\hat{\boldsymbol{u}}_{N-25,j}$ is the state of the same surrogate rollout at the fixed intermediate step $N-25$ and $\lambda_{\mathrm{aux}} = 1$. The second term adds tracking pressure before the end of the horizon and needs no stored state beyond $\boldsymbol{u}_{0,j}$. Setting $\lambda_{\mathrm{aux}} = 0$ recovers the objective used in Algorithm 2. Each surrogate uses one selected checkpoint from seed 0. Plant- and surrogate-side tracking for every PDE aggregate three independently trained policy seeds. Table 2 averages seed-specific $\log_{10}$ medians, while Appendix J pools the rollout values.

D Baseline Implementations

D.1 Reinforcement Learning: Proximal Policy Optimization

Proximal policy optimization (PPO) (Schulman et al., 2017) serves as the model-free baseline. At step k, the `controlgym` environment returns

$$r_k^{\mathrm{PPO}} = -\ell(\boldsymbol{u}_{k+1},\boldsymbol{a}_k),$$

the negative of the stage cost in Equation 1, without reward shaping or normalization. We implement PPO directly around the same `controlgym` environments used for DPC (Zhang et al., 2024). This implementation uses the same targets, actuator bounds, rollout horizons, and initial-condition partitions while allowing environment-specific PPO settings.

Policy optimization. The actor and critic receive the target-centered state $\boldsymbol{u}_k - \boldsymbol{r}$. Define $\ell_k = \ell(\boldsymbol{u}_{k+1},\boldsymbol{a}_k)$. PPO estimates advantages using generalized advantage estimation (GAE):

$$\begin{aligned}\delta_k &= -\ell_k + \gamma V_{\boldsymbol{\psi}}(\boldsymbol{u}_{k+1}-\boldsymbol{r}) - V_{\boldsymbol{\psi}}(\boldsymbol{u}_k-\boldsymbol{r}),\\ \widehat{A}_k &= \delta_k + \gamma\lambda_{\mathrm{GAE}}\widehat{A}_{k+1}.\end{aligned} \tag{15}$$

Here, γ is the discount factor, $V_{\boldsymbol{\psi}}$ is the critic with parameters $\boldsymbol{\psi}$, and λ_{GAE} is the GAE parameter. The value bootstrap is zero after the final transition of each rollout. Advantages are normalized once over the trajectories collected in a PPO iteration and then held fixed across its optimization epochs.

For actor parameters $\boldsymbol{W}$, the probability ratio between the current and behavior policies is

$$\rho_k(\boldsymbol{W}) = \frac{\pi_{\boldsymbol{W}}(\boldsymbol{a}_k \mid \boldsymbol{u}_k-\boldsymbol{r})}{\pi_{\boldsymbol{W}_{\mathrm{old}}}(\boldsymbol{a}_k \mid \boldsymbol{u}_k-\boldsymbol{r})}, \tag{16}$$

and the clipped actor loss is

$$\begin{aligned}\mathcal{L}_{\mathrm{actor}}(\boldsymbol{W}) = -\mathbb{E}_k\Big[\min\big(&\rho_k(\boldsymbol{W})\widehat{A}_k,\\ &\mathrm{clip}(\rho_k(\boldsymbol{W}),1-\epsilon_{\mathrm{clip}},1+\epsilon_{\mathrm{clip}})\,\widehat{A}_k\big)\Big].\end{aligned} \tag{17}$$

The expectation is empirical over the rollout samples collected in the current PPO iteration. The critic minimizes a squared-error loss $\mathcal{L}_V(\boldsymbol{\psi})$ against return targets constructed from the GAE estimates. The complete loss is

$$\mathcal{L}_{\mathrm{PPO}}(\boldsymbol{W}, \boldsymbol{\psi}) = \mathcal{L}_{\mathrm{actor}}(\boldsymbol{W}) + c_V \mathcal{L}_V(\boldsymbol{\psi}) - c_H \mathcal{H}(\pi_{\boldsymbol{W}}), \tag{18}$$

where c_V and c_H weight the critic and entropy terms. The reported episode return is the undiscounted sum of environment rewards. For a full-length rollout,

$$R_{\mathrm{ep}} = -\sum_{k=0}^{N-1} \ell_k = -NJ(\pi; \boldsymbol{u}_0). \tag{19}$$

The discount factor affects GAE and the value targets, but not this reported return.

Rollouts and evaluation. The PPO actor and critic each use three hidden layers of width 128. PPO also learns one state-independent Gaussian log standard deviation per action dimension, so the PPO and DPC policy-training systems are not matched in total parameter count. The actor mean uses the same $a_{\max} \tanh(\cdot)$ output transformation as the DPC policy. Training actions are sampled from the Gaussian policy and clipped to $\mathbb{A}$ before application to the plant; evaluation uses the deterministic policy mean.

Each rollout uses the environment-specific horizon N unless the environment terminates or truncates it earlier. Training initial conditions come from the training partition of $\mathcal{D}_{\mathrm{DPC}}$, and evaluation uses its held-out test partition. Table 6 reports the PPO settings. The learning rate, network size, and log-standard-deviation settings are shared across environments; the remaining listed settings are environment-specific.

Table 6: PPO hyperparameters. Shared settings are listed above the environment-specific settings.

Shared setting	Value
Learning rate	1×10^{-4}
GAE parameter λ_{GAE}	0.95
Actor and critic hidden layers	3×128
Policy log-std initialization, bounds	-1.5, $[-3.0, -0.5]$

Setting	Allen–Cahn	CDR	Burgers	Cahn–Hilliard	KdV	KS
Discount factor γ	0.95	0.95	0.95	0.97	0.95	0.97
Episodes / iteration	64	96	48	64	96	64
PPO epochs / iteration	4	3	4	4	3	4
Minibatch size	256	512	128	256	512	256
PPO clip coefficient ϵ_{clip}	0.1	0.1	0.2	0.2	0.1	0.1
Value loss coefficient c_V	0.5	0.5	1.0	0.5	0.5	0.5
Entropy coefficient c_H	0.005	0.005	0.01	0.005	0.005	0.005
Gradient norm bound	0.3	0.3	0.5	0.3	0.3	0.3

Training diagnostics. Figure 7 reports reward evolution for the recorded PPO training trace on each PDE, and Table 7 reports the corresponding CPU training time. Reward scales depend on the PDE-specific cost and rollout horizon, so values should be compared within, not across, panels. The recorded times range from 8.57 h for CDR to 36.52 h for Burgers.

D.2 Model Predictive Control Baseline

Model predictive control (MPC) provides an online optimization baseline for the five nonlinear environments. Linear CDR instead uses the LQR-controlled trajectories available in `controlgym`. Allen–Cahn and Cahn–Hilliard track the uniform reference -1; Burgers, KdV, and KS regulate to zero. The MPC internal model uses a spatial grid with $n_x^{\mathrm{MPC}} \leq n_x$, whereas closed-loop evaluation uses the full-resolution `controlgym` plant. Within this subsection, $\boldsymbol{u}_k$ denotes the restriction of the plant state to the MPC grid, and the stage cost is evaluated on that grid.

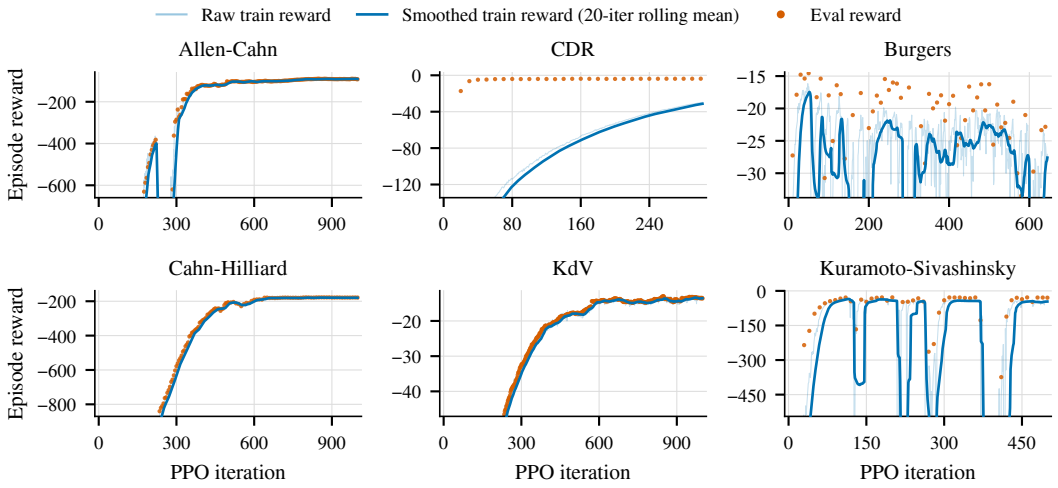

Figure 7: PPO episode reward versus training iteration for the six PDEs. Pale blue curves show raw training returns, dark blue curves show trailing 20-iteration means, and orange markers show periodic deterministic-evaluation returns. Vertical limits are panel-specific and crop some early low-return values to resolve later training dynamics.

Table 7: Recorded PPO CPU training time by PDE.

PDE	CPU training time (h)
Allen–Cahn	31.59
CDR	8.57
Burgers	36.52
Cahn–Hilliard	33.55
KdV	17.47
Kuramoto–Sivashinsky	9.64

Prediction model and online optimization. A finite-difference discretization of Equation 9, followed by a PDE-specific time integrator, defines the reduced-order one-step map

$$\boldsymbol{u}_{k+1} = \mathcal{M}_{\Delta t}(\boldsymbol{u}_k, \boldsymbol{a}_k). \tag{20}$$

Let H_{MPC} be the prediction horizon and let h index steps within an online solve. Starting from the current restricted state, MPC solves

$$\begin{aligned} \min_{\boldsymbol{a}_k,\ldots,\boldsymbol{a}_{k+H_{\mathrm{MPC}}-1}} \quad & \sum_{h=0}^{H_{\mathrm{MPC}}-1} \ell(\boldsymbol{u}_{k+h+1}, \boldsymbol{a}_{k+h}) \\ \text{s.t.} \quad & \boldsymbol{u}_{k+h+1} = \mathcal{M}_{\Delta t}(\boldsymbol{u}_{k+h}, \boldsymbol{a}_{k+h}), \\ & \boldsymbol{a}_{k+h} \in \mathbb{A}, \qquad h = 0, \ldots, H_{\mathrm{MPC}} - 1. \end{aligned} \tag{21}$$

The objective uses the stage cost ℓ from Equation 1, with the state weight resized to $0.1\boldsymbol{I}_{n_x^{\mathrm{MPC}}}$ and the action weight unchanged at $10^{-3}\boldsymbol{I}_{n_a}$. No terminal cost or state, action-rate, or terminal constraints are added. The predicted states are eliminated by recursively composing Equation 20, yielding a single-shooting problem with $n_a H_{\mathrm{MPC}}$ decision variables.

At each control step, the plant state is restricted to the MPC grid, Equation 21 is solved with CasADi/IPOPT (Andersson et al., 2019), and only the first optimized action is applied to the full-resolution plant. The procedure repeats from the next plant state. Table 8 reports the MPC-specific discretization and horizon settings. In all MPC studies, IPOPT uses a tolerance of 10^{-6} and at most 150 iterations.

MPC tractability. Figure 8 reports the online solve cost for KdV test sample 17. At the evaluation configuration, the median solve time is 6.49 s per control step. Increasing the internal grid from 64

Table 8: MPC-specific configurations for the five nonlinear environments. The interval Δt_{MPC} is the duration for which each optimized action is held; substeps are internal integration steps per interval.

PDE	n_x^{MPC}	H_{MPC}	Integrator	Substeps	Δt_{MPC}
Allen–Cahn	64	10	RK4	10	0.01
Burgers	64	5	RK4	10	0.05
Cahn–Hilliard	64	5	CN/Heun IMEX	4	0.01
KdV	64	5	RK4	50	0.01
KS ($L = 22$)	64	10	CN/Heun IMEX	10	0.5

points to the full plant resolution of 128 points raises the median to 355.05 s, a factor of 54.7. Relative to the 32-point grid, whose median is 0.205 s, the increase is a factor of 1732. With $n_x^{\mathrm{MPC}} = 64$ fixed, extending the horizon from $H_{\mathrm{MPC}} = 3$ to 20 raises the median from 2.35 s to 111.98 s, a factor of 47.7. Resolution-dependent cost growth also appears on Allen–Cahn and KS. At $H_{\mathrm{MPC}} = 10$, doubling n_x^{MPC} from 64 to 128 raises the median solve time from 2.35 to 13.55 s on Allen–Cahn, a factor of 5.77, while doubling it from 32 to 64 raises the median from 0.75 to 2.47 s on KS, a factor of 3.29. Increasing H_{MPC} from 3 to 20 raises the median from 0.23 s to 10.32 s on Allen–Cahn and from 0.15 s to 3.59 s on KS, factors of 44.9 and 24.0, respectively. Three resolution points do not support fitted scaling laws for Allen–Cahn or KS. The sweeps show why these MPC baselines use reduced internal grids where needed and finite horizons.

Across the three timing studies, actuator bounds raise the median solve time by factors of 4.60 on KdV, 4.70 on Allen–Cahn, and 7.50 on KS at matched grid and horizon settings. The corresponding unconstrained solutions exceed their actuator limits by factors of 2.71, 4.41, and 3.07, so they are not admissible alternatives. For KdV, the runs use 7, 50, 169, and 400 RK4 substeps at $n_x^{\mathrm{MPC}} = 32, 64, 96, 128$, respectively, a sequence that scales approximately cubically with n_x^{MPC}. The grid-resolution sweep therefore combines growth in nonlinear-program size and time-integration cost.

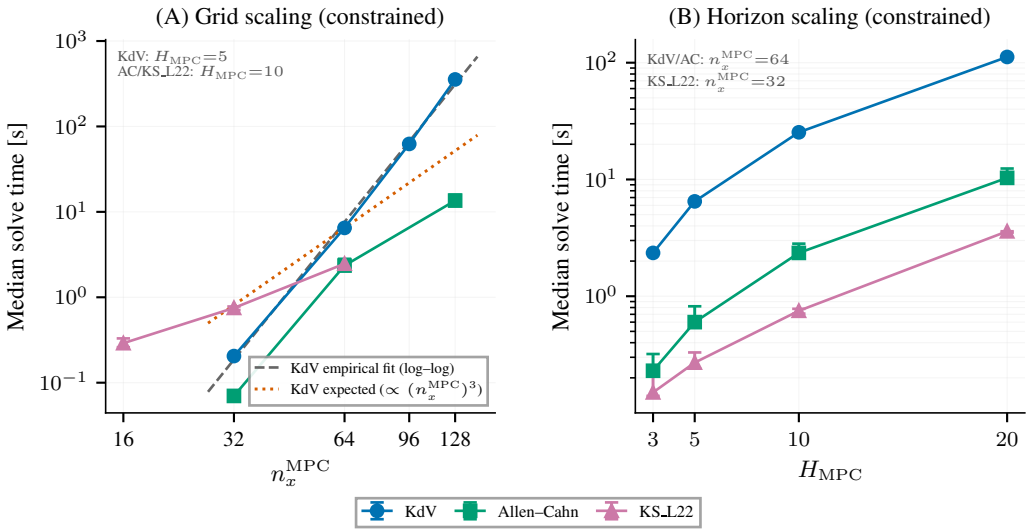

Figure 8: **MPC solve-time scaling across three PDEs.** Each point reports the median CasADi/IPOPT solve time per control step for one rollout of test sample 17; whiskers mark the interquartile range. Allen–Cahn uses 25 control steps per setting, and KdV and KS use 100. *Left:* Grid-resolution sweeps at the fixed horizons shown. The fitted trend and cubic RK4-substep reference apply only to KdV. *Right:* Prediction-horizon sweeps at the fixed grid sizes shown.

E Data Generation and Perturbed Initial Conditions

To train the dynamics surrogate $\mathcal{G}_{\boldsymbol{\theta}}$, we simulate each PDE under prescribed open-loop controls and collect state–action trajectories from the numerical solver and environment interface in `controlgym` (Zhang et al., 2024). Let $t_k = k\Delta t$. The actuator amplitudes $\boldsymbol{a}_k$ are held constant on each interval $[t_k, t_{k+1})$, and the high-fidelity solver advances the state as

$$\boldsymbol{u}_{k+1} = \mathcal{S}_{\Delta t}\left(\boldsymbol{u}_k, \boldsymbol{a}_k; \boldsymbol{\xi}_k\right), \qquad k = 0, \ldots, N-1.$$

For nonlinear PDEs, `controlgym` uses a pseudo-spectral spatial discretization (Trefethen, 1996) and ETDRK4 time integration (Cox & Matthews, 2002; Kassam & Trefethen, 2005). Linear PDEs use the corresponding discrete-time state-space representation (Zhang et al., 2024). Each trajectory pairs the simulated states with a smooth random control sequence $\boldsymbol{a}_{0:N-1,j} = (\boldsymbol{a}_{0,j}, \ldots, \boldsymbol{a}_{N-1,j})$, giving

$$\mathcal{D}_{\mathrm{dyn}} = \{(\boldsymbol{u}_{0,j}, \ldots, \boldsymbol{u}_{N,j}; \boldsymbol{a}_{0,j}, \ldots, \boldsymbol{a}_{N-1,j})\}_{j=1}^{n_{\mathrm{traj}}}. \tag{22}$$

The dataset in Equation 22 is used only to train $\mathcal{G}_{\boldsymbol{\theta}}$. Policy training instead draws initial conditions from $\mathcal{D}_{\mathrm{DPC}}$, constructed with separate random seeds from the same environment-specific families in Appendix F. The prescribed controls are sampled independently of the downstream tracking objective.

Initial-condition pools. Dynamics training uses 500 initial conditions with consecutive seeds 123–622. Each initial condition is paired with one independently sampled control trajectory from each of five frequency bands, giving $n_{\mathrm{traj}} = 2500$ trajectories. The DPC pool contains 500 independent initial-condition draws with seeds 999–1498. It supplies initial conditions for policy training and therefore has no control-frequency band structure. The two construction seed ranges are disjoint. They do not identify the held-out evaluation sets.

Prescribed controls. Let $T = N\Delta t$. For actuator $q \in \{1, \ldots, n_a\}$ and sinusoidal component $\mu \in \{1, \ldots, n_{\sin}\}$, the unnormalized multisine signal is

$$z_{q,k} = \sum_{\mu=1}^{n_{\sin}} A_{q,\mu} \sin\Big(2\pi \frac{\omega_{q,\mu}}{T} t_k + \varphi_{q,\mu}\Big), \qquad k = 0, \ldots, N-1.$$

Here, $\varphi_{q,\mu} \sim \mathcal{U}(0, 2\pi)$, $A_{q,\mu} \sim \mathcal{U}(A_{\min}, A_{\max})$, and $\omega_{q,\mu}$ is the number of cycles completed over T, sampled from the assigned frequency band. We use $n_{\sin} = 4$, $A_{\min} = 10$, and $A_{\max} = 20$ in all six environments. Each actuator signal is centered and scaled:

$$\bar{z}_q = \frac{1}{N}\sum_{k=0}^{N-1} z_{q,k}, \qquad a_{q,k} = a_{\max}\frac{z_{q,k} - \bar{z}_q}{\max_{0 \le k' < N} |z_{q,k'} - \bar{z}_q|}.$$

The normalization requires a nonzero denominator. Centering enforces zero temporal mean over the complete sampled trajectory, not over every intermediate segment.

Frequency bands. Five logarithmically spaced bands cover cycle counts from one cycle per trajectory to the Nyquist limit $\omega_{\mathrm{Nyq}} = \lfloor N/2 \rfloor$. With $\kappa_\omega = \omega_{\mathrm{Nyq}}^{1/5}$, band $\beta \in \{0, \ldots, 4\}$ samples each $\omega_{q,\mu}$ uniformly from $[\kappa_\omega^\beta, \kappa_\omega^{\beta+1}]$. Each band contains 500 trajectories.

Nominal noise settings. Process and observation noise follow the isotropic `controlgym` model,

$$\boldsymbol{\Sigma}_w = \sigma_w^2 \boldsymbol{I}_{n_x}, \qquad \boldsymbol{\Sigma}_v = \sigma_v^2 \boldsymbol{I}_{n_x}.$$

The recorded nominal variances are $\sigma_w^2 = 0$ and $\sigma_v^2 = 10^{-8}$. The recorded noise starting seeds are 10000 for dynamics training and 50000 for the DPC pool.

E.1 Perturbed initial conditions

All perturbed initial conditions are constructed from the same fixed held-out set $\{\boldsymbol{u}_{0,j}\}$. The base state indexed by j is shared across perturbation types and levels, and the random realizations are paired across levels within each stochastic construction.

Amplitude scaling. For a held-out state $\boldsymbol{u}_{0,j}$, let $\mathbf{1} \in \mathbb{R}^{n_x}$ denote the all-ones vector and define its spatial mean and scaled counterpart by

$$\bar{u}_{0,j} = \frac{1}{n_x}\sum_{i=1}^{n_x} u^i_{0,j}, \qquad \boldsymbol{u}^{\mathrm{scale}}_{0,j}(\alpha) = \bar{u}_{0,j}\mathbf{1} + \alpha\left(\boldsymbol{u}_{0,j} - \bar{u}_{0,j}\mathbf{1}\right), \quad \alpha \in \{0.5, 0.75, 1, 1.5, 2\}.$$

This transformation preserves the spatial mean and scales deviations from it. For mean-zero initial conditions, it reduces to $\boldsymbol{u}^{\mathrm{scale}}_{0,j}(\alpha) = \alpha\boldsymbol{u}_{0,j}$. The left column of Figure 9 visualizes this scaling; the right column shows the additive and burn-in constructions defined below.

Additive perturbations. We evaluate $\chi \in \{0, 10^{-2}, 5\times10^{-2}, 2\times10^{-1}\}$. Additive perturbations act directly on each held-out initial condition:

$$\boldsymbol{u}^{\mathrm{add}}_{0,j}(\chi) = \boldsymbol{u}_{0,j} + \frac{\chi\left\|\boldsymbol{u}_{0,j}\right\|_2}{\sqrt{n_x}}\boldsymbol{\epsilon}_j, \qquad \boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{n_x}).$$

For a nonzero base state, the root mean square of the relative perturbation norm equals χ, although an individual realization need not have relative norm exactly χ. Its realized displacement is

$$\widehat{\chi}^{\mathrm{add}}_j(\chi) = \frac{\left\|\boldsymbol{u}^{\mathrm{add}}_{0,j}(\chi) - \boldsymbol{u}_{0,j}\right\|_2}{\left\|\boldsymbol{u}_{0,j}\right\|_2}.$$

At $\chi = 0$, this construction returns the original held-out initial condition.

Burn-in perturbations. We use the same levels, $\chi \in \{0, 10^{-2}, 5\times10^{-2}, 2\times10^{-1}\}$. Each base state is advanced through $N_{\mathrm{burn}} = 10$ zero-control transitions. After each complete transition, an independent state disturbance $\boldsymbol{w}^{(\chi)}_{k,j} \sim \mathcal{N}\big(\mathbf{0}, \sigma^2_w(\chi)\boldsymbol{I}_{n_x}\big)$ is added, with

$$\sigma^2_w(\chi) = \frac{\chi^2\left(\mathrm{median}_j\left\|\boldsymbol{u}_{0,j}\right\|_2\right)^2}{n_x N_{\mathrm{burn}}}, \qquad \left(\sum_{k=0}^{N_{\mathrm{burn}}-1}\mathbb{E}\left\|\boldsymbol{w}^{(\chi)}_{k,j}\right\|_2^2\right)^{1/2} = \chi\,\mathrm{median}_j\left\|\boldsymbol{u}_{0,j}\right\|_2.$$

The resulting state is denoted $\boldsymbol{u}^{\mathrm{burn}}_{0,j}(\chi)$. Because the PDE dynamics propagate the injected disturbances, χ controls their aggregate scale rather than the final state displacement. We measure that displacement relative to the corresponding zero-process-noise burn-in:

$$\widehat{\chi}^{\mathrm{burn}}_j(\chi) = \frac{\left\|\boldsymbol{u}^{\mathrm{burn}}_{0,j}(\chi) - \boldsymbol{u}^{\mathrm{burn}}_{0,j}(0)\right\|_2}{\left\|\boldsymbol{u}^{\mathrm{burn}}_{0,j}(0)\right\|_2}, \qquad \widehat{\chi}^{\mathrm{burn}}(\chi) = \mathrm{median}_j\,\widehat{\chi}^{\mathrm{burn}}_j(\chi).$$

At $\chi = 0$, the state still undergoes N_{burn} zero-control transitions with zero process noise. It is therefore distinct from the original held-out state used by the additive $\chi = 0$ case. Observation noise has covariance $10^{-8}\boldsymbol{I}_{n_x}$ during burn-in, but the observations are discarded and do not affect the saved state.

F PDE Control Environments

The six environments use the periodic grid $x_i = (i-1)L/n_x$, $i = 1, \ldots, n_x$, with the discretizations in Table 1. The forcing is $f(x,t) = \sum_{q=1}^{n_a}\Phi_q(x)a_q(t)$. Below, $\mathcal{U}(a,b)$ denotes a uniform distribution and $\mathrm{sech}(z) = 1/\cosh(z)$. Initial-condition parameters are defined locally for each environment.

Allen–Cahn. With $\kappa = 10^{-2}$ and $V = 5$,

$$\begin{aligned}\partial_t u &= \kappa^2\partial_{xx}u - V(u^3 - u) + f(x,t),\\ u_0(x) &= c + (x - L/2)^2\cos\left(\frac{2\pi(x - L/2)}{L}\right),\end{aligned} \tag{23}$$

where $c \sim \mathcal{U}(-0.1, 0.1)$.

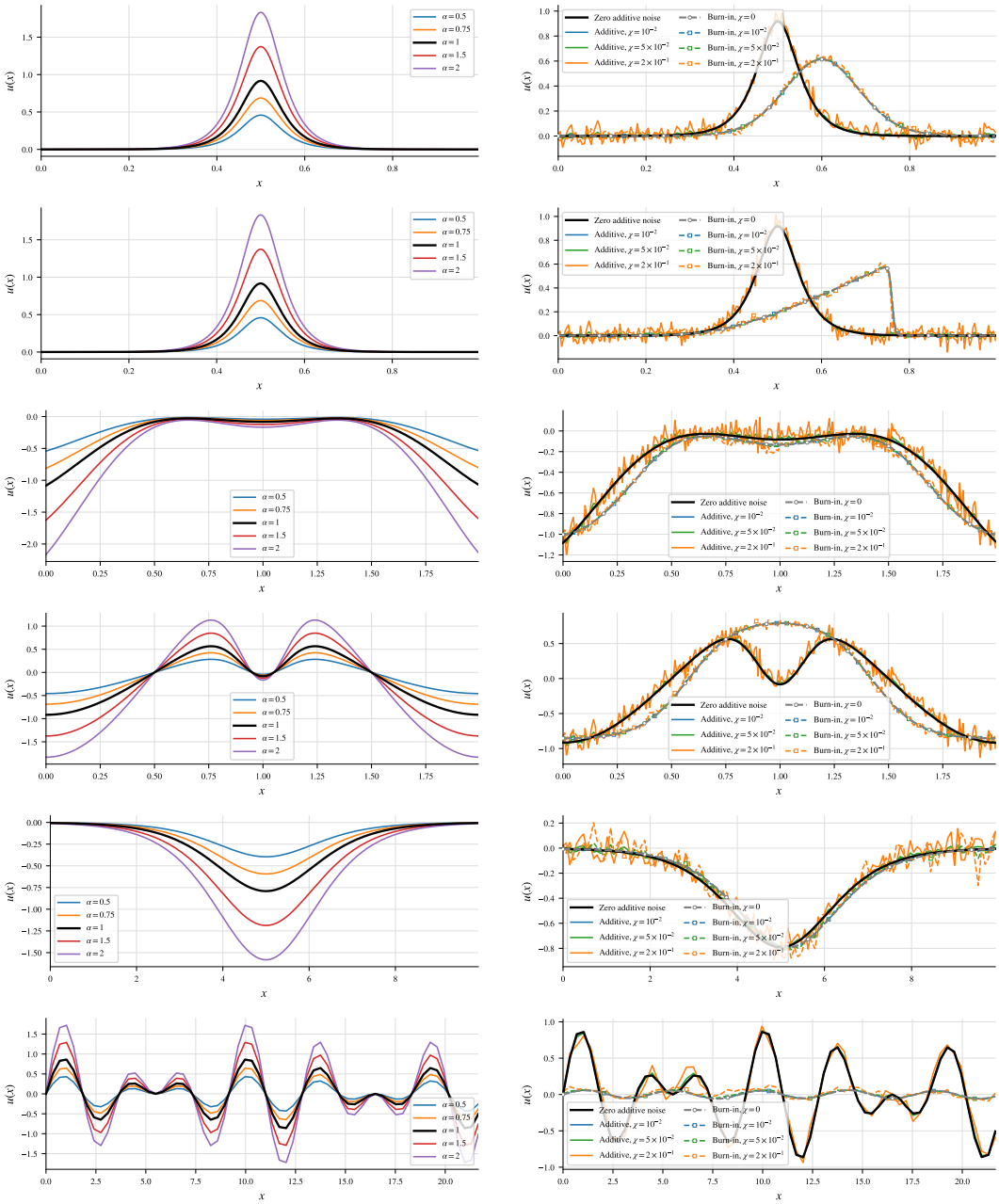

Figure 9: **Out-of-distribution initial conditions across six PDE benchmarks.** Rows correspond to the convection–diffusion–reaction, Burgers, Allen–Cahn, Cahn–Hilliard, Korteweg–de Vries, and Kuramoto–Sivashinsky equations. Both panels in each row use the same held-out sample (index 85; initial-condition seed 2086), and the horizontal axis gives physical position on the periodic domain.

Burgers. With $\kappa = 10^{-3}$,

$$\begin{aligned} \partial_t u &= -u\partial_x u + \kappa \partial_{xx} u + f(x,t), \\ u_0(x) &= A \operatorname{sech}\left(\frac{x - L/2}{s}\right), \end{aligned} \tag{24}$$

where $A \sim \mathcal{U}(0.9, 1.1)$ and $s \sim \mathcal{U}(0.04, 0.06)$.

Cahn–Hilliard. With $\kappa = 1$ and $\Gamma = 2 \times 10^{-2}$,

$$\begin{aligned} \partial_t u &= \kappa \partial_{xx}\left(u^3 - u - \Gamma \partial_{xx} u\right) + f(x,t), \\ u_0(x) &= A \cos(\pi(x - L/2)) - \exp\left[-4\left(\pi(x - L/2)\right)^2\right], \end{aligned} \tag{25}$$

where $A \sim \mathcal{U}(0.9, 1.1)$.

Convection–diffusion–reaction. With $c = 0.1$, $\kappa = 2 \times 10^{-3}$, and $\rho = 0.1$,

$$\begin{aligned} \partial_t u &= -c\partial_x u + \kappa \partial_{xx} u + \rho u + f(x,t), \\ u_0(x) &= A \operatorname{sech}\left(\frac{x - L/2}{s}\right), \end{aligned} \tag{26}$$

where $A \sim \mathcal{U}(0.9, 1.1)$ and $s \sim \mathcal{U}(0.04, 0.06)$.

Korteweg–de Vries.

$$\begin{aligned} \partial_t u &= 6u\partial_x u - \partial_{xxx} u + f(x,t), \\ u_0(x) &= -\frac{A}{2} \operatorname{sech}^2\left(\frac{\sqrt{A}}{2}(x - L/2)\right), \end{aligned} \tag{27}$$

where $A \sim \mathcal{U}(1.5, 2.5)$.

Kuramoto–Sivashinsky.

$$\begin{aligned} \partial_t u &= -u\partial_x u - \partial_{xx} u - \partial_{xxxx} u + f(x,t), \\ u_0(x) &= A \cos\left(\frac{2\pi x}{L}\right) \sin\left(\frac{12\pi x}{L}\right), \end{aligned} \tag{28}$$

where $A \sim \mathcal{U}(0.9, 1.1)$.

G PROOFS

Setup used in all proofs. A fixed policy π is run from the same initial state $\boldsymbol{u}_0$ in two worlds. On the plant, $\boldsymbol{a}_k = \pi(\boldsymbol{u}_k, \boldsymbol{r})$ and $\boldsymbol{u}_{k+1} = \mathcal{S}(\boldsymbol{u}_k, \boldsymbol{a}_k)$. On the surrogate, $\hat{\boldsymbol{a}}_k = \pi(\hat{\boldsymbol{u}}_k, \boldsymbol{r})$ and $\hat{\boldsymbol{u}}_{k+1} = \mathcal{G}_{\boldsymbol{\theta}}(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k)$ (Equation 3). Norms are Euclidean and matrix norms are the induced norms. We write $e_k = \|\boldsymbol{u}_k - \hat{\boldsymbol{u}}_k\|_2$ for the gap between the two trajectories at step k, and $\varepsilon_\pi = \varepsilon_\pi(\boldsymbol{u}_0)$ for the largest one-step surrogate error along the surrogate's own closed loop (Section 4).

G.1 TRANSFER BOUND (PROPOSITION 1)

Proposition 1 (Policy-dependent transfer bound; formal version of Equation 5)**.** *Assume that, on a set containing both trajectories, (i) the plant is Lipschitz: $\|\mathcal{S}(\boldsymbol{u},\boldsymbol{a}) - \mathcal{S}(\boldsymbol{v},\boldsymbol{b})\|_2 \le L_u\|\boldsymbol{u}-\boldsymbol{v}\|_2 + L_a\|\boldsymbol{a}-\boldsymbol{b}\|_2$; (ii) the policy is Lipschitz in the state: $\|\pi(\boldsymbol{u},\boldsymbol{r}) - \pi(\boldsymbol{v},\boldsymbol{r})\|_2 \le L_\pi\|\boldsymbol{u}-\boldsymbol{v}\|_2$; (iii) the stage cost is Lipschitz: $|\ell(\boldsymbol{u},\boldsymbol{a}) - \ell(\boldsymbol{v},\boldsymbol{b})| \le L_\ell(\|\boldsymbol{u}-\boldsymbol{v}\|_2 + \|\boldsymbol{a}-\boldsymbol{b}\|_2)$. Let $\gamma = L_u + L_a L_\pi$ and $\Sigma_N = \frac{1}{N}\sum_{k=1}^{N}\sum_{j=0}^{k-1}\gamma^j$. Then*

$$e_k \le \varepsilon_\pi \sum_{j=0}^{k-1} \gamma^j, \qquad \left|J(\pi;\boldsymbol{u}_0) - \widehat{J}(\pi;\boldsymbol{u}_0)\right| \le C_N\,\varepsilon_\pi, \quad C_N = L_\ell(1 + L_\pi)\,\Sigma_N. \tag{29}$$

Idea. The two trajectories start at the same state. At every step the surrogate adds a new error of at most ε_π, and the closed loop amplifies the existing gap by at most γ. The gap therefore grows like a geometric sum, and the cost difference is a Lipschitz function of the gap.

Proof. **Step 1 (actions follow states).** Both worlds use the same policy and the same reference, so by (ii) the action gap is controlled by the state gap: $\|\boldsymbol{a}_k - \hat{\boldsymbol{a}}_k\|_2 \le L_\pi e_k$.

Step 2 (one-step error recursion). Insert the plant evaluated at the surrogate's state and action, $\mathcal{S}(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k)$, and apply the triangle inequality:

$$e_{k+1} \le \underbrace{\|\mathcal{S}(\boldsymbol{u}_k, \boldsymbol{a}_k) - \mathcal{S}(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k)\|_2}_{\text{existing gap, propagated by the plant}} + \underbrace{\|\mathcal{S}(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k) - \mathcal{G}_{\boldsymbol{\theta}}(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k)\|_2}_{\text{new surrogate error at step } k}.$$

By (i) and Step 1, the first term is at most $L_u e_k + L_a L_\pi e_k = \gamma e_k$. The second term is a one-step surrogate error on the surrogate's own trajectory, so it is at most ε_π by definition. Hence $e_{k+1} \le \gamma e_k + \varepsilon_\pi$.

Step 3 (unroll the recursion). Let $s_k = \sum_{j=0}^{k-1} \gamma^j$, with $s_0 = 0$, so that $s_{k+1} = 1 + \gamma s_k$. Since $e_0 = 0 = \varepsilon_\pi s_0$, and $e_k \le \varepsilon_\pi s_k$ implies $e_{k+1} \le \gamma \varepsilon_\pi s_k + \varepsilon_\pi = \varepsilon_\pi s_{k+1}$, induction gives $e_k \le \varepsilon_\pi s_k$ for all k. This is the first claim. For example, $e_1 \le \varepsilon_\pi$, $e_2 \le (1+\gamma)\varepsilon_\pi$, and $e_3 \le (1+\gamma+\gamma^2)\varepsilon_\pi$.

Step 4 (from states to cost). By (iii) and Step 1, each stage cost differs by at most $L_\ell(e_{k+1} + L_\pi e_k)$. Since $s_k \le s_{k+1}$, this is at most $L_\ell(1 + L_\pi)\varepsilon_\pi s_{k+1}$. Averaging over the N steps,

$$\begin{aligned} \left|J - \widehat{J}\right| &\le \frac{1}{N} \sum_{k=0}^{N-1} \left|\ell(\boldsymbol{u}_{k+1}, \boldsymbol{a}_k) - \ell(\hat{\boldsymbol{u}}_{k+1}, \hat{\boldsymbol{a}}_k)\right| \\ &\le \frac{L_\ell(1 + L_\pi)\,\varepsilon_\pi}{N} \sum_{k=1}^{N} s_k = C_N\,\varepsilon_\pi. \qquad \square \end{aligned}$$

Remarks. (a) The state bound is tight: for a scalar linear plant $u_{k+1} = L_u u_k + L_a a_k$, policy $a_k = L_\pi u_k$, and a surrogate that adds a constant ε_π at every step, Step 2 holds with equality. (b) If $\gamma < 1$ the closed loop contracts and C_N stays bounded as N grows. If $\gamma > 1$, C_N can grow exponentially in N, so the bound is informative only over short horizons. (c) The quadratic stage cost is Lipschitz only on bounded sets. The proposition assumes that such a set contains both trajectories; it does not infer bounded states from bounded actuation.

G.2 Policy-performance bound

Proposition 1 concerns one policy evaluated in two worlds. Here we use it to compare the trained policy with any other policy on the plant. Let $\mathcal{J}(\pi) = \mathbb{E}_{\boldsymbol{u}_0 \sim \mathcal{D}_{\mathrm{DPC}}}[J(\pi; \boldsymbol{u}_0)]$ and $\widehat{\mathcal{J}}(\pi) = \mathbb{E}_{\boldsymbol{u}_0 \sim \mathcal{D}_{\mathrm{DPC}}}[\widehat{J}(\pi; \boldsymbol{u}_0)]$ be the expected plant and surrogate costs, assumed finite. For a policy class $\mathcal{P}$ containing the trained policy $\pi_{\boldsymbol{W}^\star}$, define

$$\delta_{\mathrm{opt}} = \widehat{\mathcal{J}}(\pi_{\boldsymbol{W}^\star}) - \inf_{\pi \in \mathcal{P}} \widehat{\mathcal{J}}(\pi) \ \ge 0 \quad \text{(gap to the best policy on the surrogate)},$$

$$\Delta(\pi) = \left|\mathcal{J}(\pi) - \widehat{\mathcal{J}}(\pi)\right| \quad \text{(how much the surrogate misjudges policy } \pi\text{)}.$$

Idea. To go from the plant cost of $\pi_{\boldsymbol{W}^\star}$ to the plant cost of a comparator π°, pass through the surrogate: move to the surrogate, compare the two policies there, and move back.

Proof. For any $\pi^\circ \in \mathcal{P}$, adding and subtracting surrogate costs gives the exact identity

$$\mathcal{J}(\pi_{\boldsymbol{W}^\star}) - \mathcal{J}(\pi^\circ) = \underbrace{\left[\mathcal{J}(\pi_{\boldsymbol{W}^\star}) - \widehat{\mathcal{J}}(\pi_{\boldsymbol{W}^\star})\right]}_{\le \Delta(\pi_{\boldsymbol{W}^\star})} + \underbrace{\left[\widehat{\mathcal{J}}(\pi_{\boldsymbol{W}^\star}) - \widehat{\mathcal{J}}(\pi^\circ)\right]}_{\le \delta_{\mathrm{opt}}} + \underbrace{\left[\widehat{\mathcal{J}}(\pi^\circ) - \mathcal{J}(\pi^\circ)\right]}_{\le \Delta(\pi^\circ)}.$$

The middle bound holds because $\widehat{\mathcal{J}}(\pi^\circ) \ge \inf_{\pi \in \mathcal{P}} \widehat{\mathcal{J}}(\pi)$. Therefore

$$\mathcal{J}(\pi_{\boldsymbol{W}^\star}) - \mathcal{J}(\pi^\circ) \le \Delta(\pi_{\boldsymbol{W}^\star}) + \delta_{\mathrm{opt}} + \Delta(\pi^\circ). \tag{30}$$

Finally, $\Delta(\pi) \le \mathbb{E}_{\boldsymbol{u}_0}\left[\left|J(\pi; \boldsymbol{u}_0) - \widehat{J}(\pi; \boldsymbol{u}_0)\right|\right]$, so if the assumptions of Proposition 1 hold with constants that do not depend on $\boldsymbol{u}_0$, then $\Delta(\pi) \le C_N\, \mathbb{E}_{\boldsymbol{u}_0}[\varepsilon_\pi(\boldsymbol{u}_0)]$. $\square$

Remarks. The bound is applied separately to $\pi_{\boldsymbol{W}^\star}$ and π°, each with its own Lipschitz constant L_π and its own ε_π. In words: the trained policy is nearly as good on the plant as any comparator if (1) training nearly minimizes the surrogate cost and (2) the surrogate is accurate on the trajectories of both policies. When training includes the auxiliary term in Equation 14, δ_{opt} refers to the primary objective $\widehat{\mathcal{J}}$, not to $\mathcal{L}_{\mathrm{train}}$.

G.3 Forward accuracy versus gradient accuracy (Proposition 2)

Proposition 2 (Vanishing forward error with persistent gradient error)**.** *Let $N = 1$, and let $\mathcal{S}$, $\pi_{\boldsymbol{W}}$, and ℓ be continuously differentiable. Fix $\boldsymbol{u}_0$, $\boldsymbol{r}$, and anchor parameters $\bar{\boldsymbol{W}}$ with anchor action $\bar{\boldsymbol{a}} = \pi_{\bar{\boldsymbol{W}}}(\boldsymbol{u}_0, \boldsymbol{r})$. For $c, \eta > 0$ and unit vectors $\boldsymbol{e} \in \mathbb{R}^{n_a}$, $\boldsymbol{\phi} \in \mathbb{R}^{n_x}$, define the surrogate*

$$\mathcal{G}_\eta(\boldsymbol{u}, \boldsymbol{a}) = \mathcal{S}(\boldsymbol{u}, \boldsymbol{a}) + \eta \sin\big(c\langle \boldsymbol{a} - \bar{\boldsymbol{a}}, \boldsymbol{e}\rangle/\eta\big)\, \boldsymbol{\phi},$$

and let $J(\boldsymbol{W})$ and $\widehat{J}_\eta(\boldsymbol{W})$ be the one-step costs on $\mathcal{S}$ and $\mathcal{G}_\eta$. Then

$$\begin{gathered} \sup_{\boldsymbol{u},\boldsymbol{a}} \|\mathcal{G}_\eta - \mathcal{S}\|_2 \le \eta, \qquad \sup_{\boldsymbol{u},\boldsymbol{a}} \|\partial_{\boldsymbol{a}}\mathcal{G}_\eta - \partial_{\boldsymbol{a}}\mathcal{S}\|_2 = c, \\ \nabla_{\boldsymbol{W}} \widehat{J}_\eta(\bar{\boldsymbol{W}}) - \nabla_{\boldsymbol{W}} J(\bar{\boldsymbol{W}}) = c\, \langle \boldsymbol{\phi}, \nabla_{\boldsymbol{u}}\ell(\boldsymbol{u}_1, \bar{\boldsymbol{a}})\rangle\, \bar{\boldsymbol{B}}^\top \boldsymbol{e}, \end{gathered} \tag{31}$$

where $\boldsymbol{u}_1 = \mathcal{S}(\boldsymbol{u}_0, \bar{\boldsymbol{a}})$ and $\bar{\boldsymbol{B}} = \partial_{\boldsymbol{W}}\pi_{\boldsymbol{W}}(\boldsymbol{u}_0, \boldsymbol{r})|_{\boldsymbol{W}=\bar{\boldsymbol{W}}}$. The gradient error does not depend on η and is nonzero whenever $\langle \boldsymbol{\phi}, \nabla_{\boldsymbol{u}}\ell\rangle \neq 0$ and $\bar{\boldsymbol{B}}^\top \boldsymbol{e} \neq \boldsymbol{0}$.

Idea. The surrogate is the true plant plus a small, fast ripple of height η and slope c. As $\eta \to 0$ the ripple becomes invisible in the output but it stays equally steep, because it oscillates faster. Policy training uses the slope, not the height, so the gradient error does not shrink.

Proof. The anchor action $\bar{\boldsymbol{a}}$ is a fixed constant in the definition of $\mathcal{G}_\eta$; it is not differentiated with respect to $\boldsymbol{W}$.

Step 1 (the ripple is small). Since $|\sin(\cdot)| \le 1$ and $\|\boldsymbol{\phi}\|_2 = 1$, we have $\|\mathcal{G}_\eta(\boldsymbol{u}, \boldsymbol{a}) - \mathcal{S}(\boldsymbol{u}, \boldsymbol{a})\|_2 \le \eta$ for all $(\boldsymbol{u}, \boldsymbol{a})$.

Step 2 (the ripple is steep). Differentiating in the action, the factor η cancels with the $1/\eta$ inside the sine:

$$\partial_{\boldsymbol{a}}\mathcal{G}_\eta - \partial_{\boldsymbol{a}}\mathcal{S} = c\cos\big(c\langle \boldsymbol{a} - \bar{\boldsymbol{a}}, \boldsymbol{e}\rangle/\eta\big)\, \boldsymbol{\phi}\boldsymbol{e}^\top.$$

The rank-one matrix $\boldsymbol{\phi}\boldsymbol{e}^\top$ has norm $\|\boldsymbol{\phi}\|_2\|\boldsymbol{e}\|_2 = 1$, so the Jacobian error is at most c everywhere and equals c at $\boldsymbol{a} = \bar{\boldsymbol{a}}$, where the cosine is one. This proves the first two relations in Equation 31.

Step 3 (both models agree at the anchor). At $\boldsymbol{a} = \bar{\boldsymbol{a}}$ the sine is zero, so $\mathcal{G}_\eta(\boldsymbol{u}_0, \bar{\boldsymbol{a}}) = \mathcal{S}(\boldsymbol{u}_0, \bar{\boldsymbol{a}}) = \boldsymbol{u}_1$. Both models therefore produce the same next state and the same cost at $\bar{\boldsymbol{W}}$.

Step 4 (only the model Jacobian differs in the gradient). With $N = 1$, the cost is $\ell(\boldsymbol{u}_1, \boldsymbol{a}_0)$ with $\boldsymbol{a}_0 = \pi_{\boldsymbol{W}}(\boldsymbol{u}_0, \boldsymbol{r})$. By the chain rule, the plant gradient at $\bar{\boldsymbol{W}}$ is

$$\nabla_{\boldsymbol{W}} J(\bar{\boldsymbol{W}}) = \bar{\boldsymbol{B}}^\top \Big[\underbrace{(\partial_{\boldsymbol{a}}\mathcal{S})^\top \nabla_{\boldsymbol{u}}\ell(\boldsymbol{u}_1, \bar{\boldsymbol{a}})}_{\text{through the dynamics}} + \underbrace{\nabla_{\boldsymbol{a}}\ell(\boldsymbol{u}_1, \bar{\boldsymbol{a}})}_{\text{direct action cost}}\Big],$$

with $\partial_{\boldsymbol{a}}\mathcal{S}$ evaluated at $(\boldsymbol{u}_0, \bar{\boldsymbol{a}})$. The surrogate gradient has the same form with $\partial_{\boldsymbol{a}}\mathcal{S}$ replaced by $\partial_{\boldsymbol{a}}\mathcal{G}_\eta$. By Step 3, the cost derivatives $\nabla_{\boldsymbol{u}}\ell$ and $\nabla_{\boldsymbol{a}}\ell$ are evaluated at the same point in both gradients, so the direct terms cancel and only the Jacobians differ. Using Step 2 at $\boldsymbol{a} = \bar{\boldsymbol{a}}$,

$$\nabla_{\boldsymbol{W}} \widehat{J}_\eta(\bar{\boldsymbol{W}}) - \nabla_{\boldsymbol{W}} J(\bar{\boldsymbol{W}}) = \bar{\boldsymbol{B}}^\top \big(c\, \boldsymbol{\phi}\boldsymbol{e}^\top\big)^\top \nabla_{\boldsymbol{u}}\ell(\boldsymbol{u}_1, \bar{\boldsymbol{a}}) = c\, \langle \boldsymbol{\phi}, \nabla_{\boldsymbol{u}}\ell(\boldsymbol{u}_1, \bar{\boldsymbol{a}})\rangle\, \bar{\boldsymbol{B}}^\top \boldsymbol{e}.$$

No η appears on the right-hand side, so the gradient error stays fixed as the forward error $\eta \to 0$. □

Extension to fixed forecast horizons. The same surrogate also has vanishing multi-step forecast error. Run the plant and $\mathcal{G}_\eta$ open loop from the same state under the same prescribed actions, and suppose $\mathcal{S}$ is L_u-Lipschitz in its state on a set containing both rollouts, with L_u independent of η. Repeating Steps 2 and 3 of the proof of Proposition 1 with $L_\pi = 0$ and ε_π replaced by η gives $e_{k+1} \le L_u e_k + \eta$, so $e_H \le \eta \sum_{j=0}^{H-1} L_u^j = O(\eta)$ for any fixed horizon H, and the endpoint MSE is $O(\eta^2)$. Relative errors behave the same way if the reference-state norms stay bounded away from zero.

Table 9: Ratio ρ of surrogate-side to plant-side median terminal tracking error, Equation 8, formed from the median columns of Tables 12 and 11. Values below one indicate optimistic surrogate evaluation.

PDE	TI-DON	RNO	FNO_1	FNO_5
AC	12.9	5.39	17.5	0.901
Burgers	3.26	0.185	2.20	1.01
CH	1.00	0.0117	0.0727	1.45
CDR	73.8	12.3	0.932	1.08
KdV	0.281	0.212	2.16	1.05
KS	0.188	0.285	0.263	0.597

When forward accuracy does control the Jacobian. The construction in Proposition 2 relies on curvature that grows without bound, since the second derivative of the ripple in the action is of order c^2/η. Bounding that curvature restores the implication. Let $f = \mathcal{G}_{\boldsymbol{\theta}} - \mathcal{S}$ and suppose $\sup \|f\|_2 \le \varepsilon$ and $\sup \|\partial_{\boldsymbol{a}}^2 f\|_2 \le M_2$ on the region of interest. For unit vectors $\boldsymbol{e}$ and $\boldsymbol{\phi}$, the scalar function $g(t) = \langle \boldsymbol{\phi}, f(\boldsymbol{u}, \boldsymbol{a} + t\boldsymbol{e})\rangle$ satisfies $|g| \le \varepsilon$ and $|g''| \le M_2$, so the Landau–Kolmogorov inequality on the real line gives $|g'(0)| \le \sqrt{2\varepsilon M_2}$. Taking suprema over $\boldsymbol{e}$ and $\boldsymbol{\phi}$,

$$\left\|\partial_{\boldsymbol{a}}\mathcal{G}_{\boldsymbol{\theta}} - \partial_{\boldsymbol{a}}\mathcal{S}\right\|_2 \le \sqrt{2\,\varepsilon\, M_2}. \tag{32}$$

The counterexample is tight for this bound up to a constant, since substituting $\varepsilon = \eta$ and $M_2 = c^2/\eta$ returns $\sqrt{2}\,c$ against an actual Jacobian error of c. A parallel statement holds when the discrepancy is band-limited in a variable: if f restricted to a line in that variable is a trigonometric polynomial of degree at most K, Bernstein's inequality gives $\|\partial f\| \le K\|f\|$, so the Jacobian error is at most $K\varepsilon$. Spectral truncation in an FNO limits the spatial frequencies of f and therefore bounds the state Jacobian error in terms of the forward error, while the dependence on the actuation passes through the nonlinear lifting and is not band-limited by mode truncation, so M_2 remains the operative quantity for the action Jacobian that DPC differentiates. We do not measure M_2 for the trained surrogates; doing so would turn Equation 32 into a usable certificate.

Interpretation. The gradient error is the product of three factors: the Jacobian error c, the sensitivity of the cost to the state along $\boldsymbol{\phi}$, and the ability of the policy parameters to move the action along $\boldsymbol{e}$. It vanishes if either sensitivity is zero. The surrogate is built around a chosen anchor, so the result is a counterexample: it shows that forward accuracy alone cannot guarantee accurate policy gradients. It does not claim that every Jacobian error makes policy optimization fail.

H Derived Quantities

For the reported quantities, let $\pi_{m,s}$ denote the policy trained through surrogate configuration m with policy seed s, and let $\rho_{m,s} = \rho(\pi_{m,s})$ be its ratio from Equation 8. To match the aggregation in Table 2, Table 9 reports

$$\rho_m = \left(\prod_{s=1}^{3} \rho_{m,s}\right)^{1/3}. \tag{33}$$

Thus, ρ_m is the geometric mean of three matched seed-specific ratios, equivalently the ratio of the geometric means of the surrogate-side and plant-side seed medians. It is not the ratio of the pooled medians in Tables 11 and 12.

Selecting the lowest-error forecaster. Table 10 compares the controller associated with the lowest median forecasting error against the lowest plant-side tracking error among all four surrogate configurations. The forecast winner is the tracking winner on four benchmarks; its tracking error is 4.2 times the best on CH and 1.07 times the best on KdV.

Table 10: Plant-side tracking error of the forecast winner divided by the lowest among the four surrogate configurations. Each error is the geometric mean of three policy-seed medians, as in Table 2.

PDE	Forecast winner	Tracking-error ratio
AC	FNO_5	1.00
Burgers	FNO_5	1.00
CH	FNO_1	4.20
CDR	FNO_5	1.00
KdV	FNO_5	1.07
KS	FNO_5	1.00

I Closed-Loop Diagnostic Metrics

For a frozen surrogate and its trained policy, let $(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k)$ denote the state and action on the surrogate closed loop in Equation 3. We evaluate the surrogate and plant at these same state–action pairs to separate errors in transition values from errors in their local sensitivities.

On-policy one-step defect. The defect at step k and its trajectory maximum are

$$d_k = \left\| \mathcal{G}_{\boldsymbol{\theta}}(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k) - \mathcal{S}(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k) \right\|_2, \qquad \varepsilon_\pi(\boldsymbol{u}_0) = \max_{0 \le k < N} d_k. \tag{34}$$

The maximum is the quantity in Proposition 1; the evolution of d_k provides additional temporal information, distinguishing isolated errors from persistent mismatch along the rollout.

Local action sensitivity. Write $\boldsymbol{J}^{\mathcal{G}}_{a,k} = \partial_{\boldsymbol{a}} \mathcal{G}_{\boldsymbol{\theta}}$ and $\boldsymbol{J}^{\mathcal{S}}_{a,k} = \partial_{\boldsymbol{a}} \mathcal{S}$ at $(\hat{\boldsymbol{u}}_k, \hat{\boldsymbol{a}}_k)$. Their relative mismatch is

$$E_a(k) = \frac{\left\| \boldsymbol{J}^{\mathcal{G}}_{a,k} - \boldsymbol{J}^{\mathcal{S}}_{a,k} \right\|_F}{\left\| \boldsymbol{J}^{\mathcal{S}}_{a,k} \right\|_F + \epsilon}, \qquad \epsilon > 0. \tag{35}$$

The derivatives are taken with respect to the eight actuator amplitudes, rather than the spatial forcing field. The fixed amplitude-to-forcing map is therefore included within both derivatives. We use the same actuator lift for the surrogate and plant and verify this correspondence before evaluating $E_a(k)$. Thus, d_k measures the total transition mismatch, while $E_a(k)$ isolates disagreement in the local response to actuator amplitudes.

Local state sensitivity. For fixed unit-norm state directions $\{\boldsymbol{v}_r\}$, define $\boldsymbol{J}^{\mathcal{G}}_{u,k} = \partial_{\boldsymbol{u}} \mathcal{G}_{\boldsymbol{\theta}}$ and $\boldsymbol{J}^{\mathcal{S}}_{u,k} = \partial_{\boldsymbol{u}} \mathcal{S}$ at the same pair. The probed relative mismatch is

$$E_u^{\mathrm{probe}}(k) = \left[\frac{\sum_r \left\| (\boldsymbol{J}^{\mathcal{G}}_{u,k} - \boldsymbol{J}^{\mathcal{S}}_{u,k}) \boldsymbol{v}_r \right\|_2^2}{\sum_r \left\| \boldsymbol{J}^{\mathcal{S}}_{u,k} \boldsymbol{v}_r \right\|_2^2 + \epsilon} \right]^{1/2}. \tag{36}$$

This quantity tests Jacobian agreement only along the selected probe directions. With Fourier probes restricted to modes $0, \ldots, k_{\max}$, it is insensitive to mismatch confined to higher modes. A small E_u^{probe} therefore establishes agreement only on the probed subspace.

This distinction is relevant to DPC because the state Jacobian propagates closed-loop sensitivities through the rollout in Equation 4. Errors along directions that contribute strongly to this propagation can distort the resulting policy gradient. Conversely, a large E_u^{probe} establishes local sensitivity mismatch in the tested directions, but does not by itself show that the mismatch caused poor control.

What the metrics measure. The diagnostics distinguish transition-value agreement from first-order agreement. The defect d_k compares the transition values, while $E_a(k)$ and $E_u^{\mathrm{probe}}(k)$ compare their derivatives with respect to action and state. Small one-step value error does not imply accurate derivatives. Proposition 2 makes this distinction explicit by constructing maps whose value error vanishes while their derivative mismatch remains nonzero.

Propagation of local defects. The quantities d_k, $E_a(k)$, and $E_u^{\mathrm{probe}}(k)$ are local, one-step diagnostics. Let $e_k = \|\boldsymbol{u}_k - \hat{\boldsymbol{u}}_k\|_2$, with $e_0 = 0$, and let γ_k be a local Lipschitz factor of the plant closed-loop map $\boldsymbol{u} \mapsto \mathcal{S}(\boldsymbol{u}, \pi(\boldsymbol{u}, \boldsymbol{r}))$ over the region between the two trajectories at step k. Then the one-step recursion and its unrolling give

$$e_{k+1} \le \gamma_k e_k + d_k, \qquad e_k \le \sum_{j=0}^{k-1} \left(\prod_{i=j+1}^{k-1} \gamma_i \right) d_j, \tag{37}$$

where an empty product equals one; if $\gamma_k \le \gamma$ and $d_k \le \varepsilon_\pi$ along the rollout, Equation 37 reduces to $e_k \le \varepsilon_\pi \sum_{j=0}^{k-1} \gamma^j$, the state-error bound in Proposition 1. At a differentiable point, the corresponding local closed-loop linearization is $\boldsymbol{J}^{\mathcal{S}}_{u,k} + \boldsymbol{J}^{\mathcal{S}}_{a,k}\boldsymbol{\Pi}_{u,k}$. Thus, contracting gains suppress past value defects and derivative errors, whereas expansive gains amplify them. The same local Jacobian factors govern sensitivity propagation through the rollout, which makes derivative mismatch particularly consequential along strongly amplified directions.

Diagnostic interpretation. The diagnostics identify different forms of surrogate–plant mismatch along the visited closed loop. Large d_k together with large $E_a(k)$ or $E_u^{\mathrm{probe}}(k)$ indicates disagreement in both transition values and local sensitivities. Small ε_π with large Jacobian mismatch is consistent with a sensitivity-based explanation for poor transfer, whereas large ε_π with small measured Jacobian mismatch points more directly to transition-value error along the rollout.

If all three diagnostics are small while plant-side control remains poor, the measured local discrepancies do not explain the failure. Possible sources include mismatch outside the probed state subspace, higher-order effects, or limitations of the learned policy or its optimization. These diagnostics establish associations along a trained closed loop; none individually identifies a causal source of poor transfer.

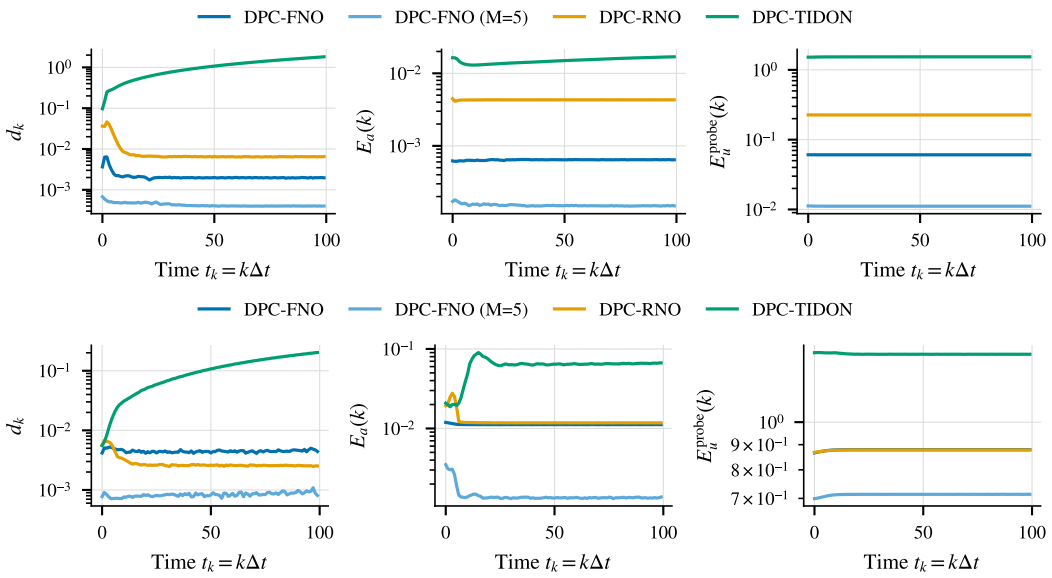

Figure 10: **Closed-loop sensitivity diagnostics along the surrogate rollout.** *Top:* CDR, evaluated on 16 held-out initial conditions. *Bottom:* KdV, evaluated on 4 held-out initial conditions. We measure the one-step on-policy defect d_k, the action-Jacobian mismatch $E_a(k)$, and the probed state-Jacobian mismatch $E_u^{\mathrm{probe}}(k)$ at $t_k = k\Delta t$, with $N\Delta t = T$.

J Full Evaluation Statistics

Tables 11 and 12 pool 150 endpoint MSE values per DPC configuration: three independently trained policy seeds evaluated on 50 initial conditions each. Their statistics therefore combine policy-seed

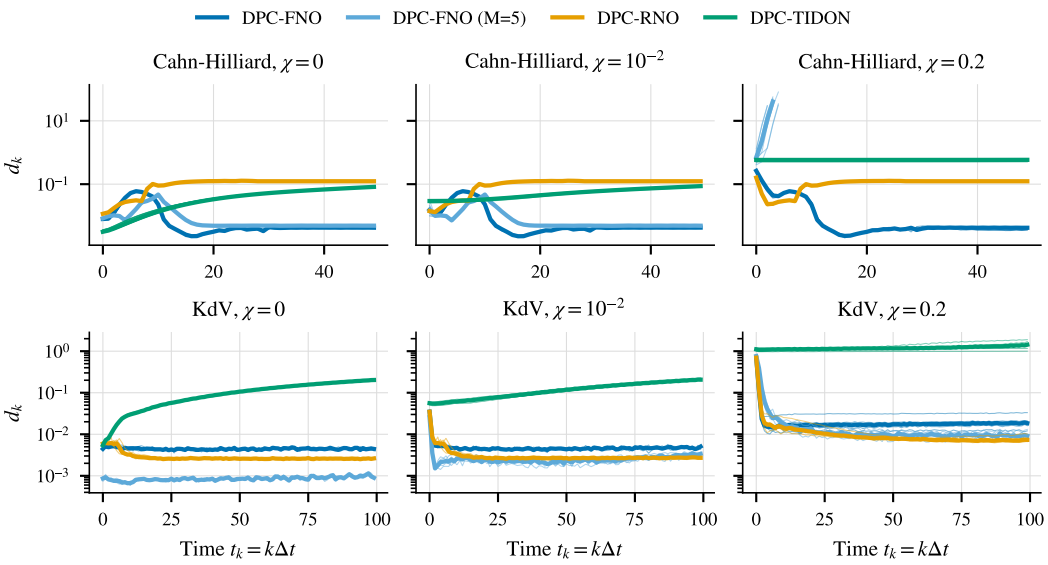

Figure 11: One-step on-policy defect d_k along the surrogate rollout, $t_k = k\Delta t$ with $N\Delta t = T$, for Cahn–Hilliard (CH, top) and KdV (bottom) under burn-in perturbations. Each initial state first evolves for 10 uncontrolled steps under process noise of level χ (columns: $\chi = 0, 10^{-2}, 0.2$). Thin lines show individual initial conditions and bold lines their median (16 for CH, 4 for KdV); the y-axis is shared across each row. Under $\chi = 0.2$, CH DPC-FNO ($M = 5$) surrogate rollout diverges within the first few steps, so the corresponding curves terminate early.

and initial-condition variation. The PPO rows use 50 evaluations from one policy seed, and Table 13 reports 50 open-loop evaluations per surrogate configuration. Figures 12 and 13 use the same held-out initial conditions. Figure 12 reports cumulative reward, $-NJ(\pi; \boldsymbol{u}_0)$, using the same $\boldsymbol{Q}$ and $\boldsymbol{R}$ across methods, while Figure 13 reports plant roll-out wall-clock time. AC and CH track the uniform target -1, while the remaining environments track zero. Table entries use scientific notation, with $ae b$ denoting $a \times 10^b$.

Table 11: Plant-side closed-loop endpoint tracking MSE.

PDE	Model	Mean ± std	Median $[q_{25}, q_{75}]$	Maximum
AC	TI-DON	9.33e-5 ± 4.44e-6	9.57e-5 [8.71e-5, 9.71e-5]	9.71e-5
AC	RNO	1.96e-7 ± 3.53e-8	1.80e-7 [1.63e-7, 2.45e-7]	2.45e-7
AC	FNO_1	2.11e-7 ± 6.69e-8	1.70e-7 [1.59e-7, 3.06e-7]	3.06e-7
AC	FNO_5	1.64e-7 ± 1.14e-7	1.21e-7 [5.17e-8, 3.20e-7]	3.20e-7
AC	PPO	6.66e-4 ± 3.86e-8	6.66e-4 [6.66e-4, 6.66e-4]	6.66e-4
Burgers	TI-DON	4.25e-5 ± 5.97e-6	4.36e-5 [3.52e-5, 4.86e-5]	4.99e-5
Burgers	RNO	7.73e-6 ± 1.95e-6	8.23e-6 [5.24e-6, 9.59e-6]	1.15e-5
Burgers	FNO_1	8.13e-7 ± 8.04e-8	8.23e-7 [7.57e-7, 8.73e-7]	9.55e-7
Burgers	FNO_5	7.11e-7 ± 1.16e-7	7.08e-7 [6.33e-7, 8.07e-7]	1.01e-6
Burgers	PPO	1.31e-5 ± 1.41e-7	1.31e-5 [1.30e-5, 1.32e-5]	1.35e-5
CH	TI-DON	1.06e0 ± 7.90e-2	1.01e0 [9.99e-1, 1.17e0]	1.17e0
CH	RNO	6.85e-4 ± 2.29e-4	7.93e-4 [3.68e-4, 8.95e-4]	8.95e-4
CH	FNO_1	1.96e-6 ± 8.26e-7	2.27e-6 [8.28e-7, 2.76e-6]	2.77e-6
CH	FNO_5	4.16e-7 ± 5.77e-8	4.21e-7 [3.43e-7, 4.83e-7]	4.85e-7
CH	PPO	1.67e-5 ± 5.22e-8	1.67e-5 [1.67e-5, 1.67e-5]	1.68e-5
CDR	TI-DON	3.00e-4 ± 7.07e-5	2.63e-4 [2.37e-4, 3.98e-4]	3.98e-4
CDR	RNO	9.19e-8 ± 1.70e-8	8.05e-8 [7.93e-8, 1.16e-7]	1.16e-7
CDR	FNO_1	8.93e-8 ± 1.26e-8	8.59e-8 [7.59e-8, 1.06e-7]	1.06e-7
CDR	FNO_5	5.33e-8 ± 8.55e-9	5.19e-8 [4.36e-8, 6.43e-8]	6.43e-8
CDR	PPO	3.15e-5 ± 7.03e-8	3.16e-5 [3.15e-5, 3.16e-5]	3.17e-5
KdV	TI-DON	7.67e-4 ± 9.99e-5	8.12e-4 [6.30e-4, 8.55e-4]	8.72e-4
KdV	RNO	3.85e-5 ± 4.01e-6	4.03e-5 [3.39e-5, 4.19e-5]	4.49e-5
KdV	FNO_1	5.78e-6 ± 1.00e-6	5.75e-6 [5.08e-6, 6.41e-6]	8.14e-6
KdV	FNO_5	6.11e-6 ± 7.90e-7	6.11e-6 [5.37e-6, 6.52e-6]	8.24e-6
KdV	PPO	2.27e-4 ± 3.60e-5	2.05e-4 [1.99e-4, 2.56e-4]	2.99e-4

PDE	Model	Mean ± std	Median [q_{25}, q_{75}]	Maximum
KS	TI-DON	4.91e-3 ± 6.60e-4	4.51e-3 [4.39e-3, 5.82e-3]	5.86e-3
KS	RNO	1.66e-4 ± 4.10e-5	1.42e-4 [1.39e-4, 2.15e-4]	2.43e-4
KS	FNO_1	4.05e-5 ± 4.40e-6	3.99e-5 [3.75e-5, 4.29e-5]	5.75e-5
KS	FNO_5	8.57e-6 ± 2.78e-6	8.02e-6 [6.37e-6, 1.04e-5]	1.68e-5
KS	PPO	7.06e-5 ± 7.70e-8	7.06e-5 [7.06e-5, 7.06e-5]	7.07e-5

Table 12: Surrogate-side closed-loop endpoint tracking MSE. Each surrogate evaluates the policy trained through it.

PDE	Model	Mean ± std	Median [q_{25}, q_{75}]	Maximum
AC	TI-DON	1.23e-3 ± 1.96e-5	1.23e-3 [1.21e-3, 1.25e-3]	1.27e-3
AC	RNO	9.18e-7 ± 2.76e-7	9.70e-7 [5.79e-7, 1.18e-6]	1.46e-6
AC	FNO_1	2.94e-6 ± 1.99e-7	2.98e-6 [2.85e-6, 3.06e-6]	3.46e-6
AC	FNO_5	1.53e-7 ± 9.04e-8	1.09e-7 [7.51e-8, 2.75e-7]	2.93e-7
Burgers	TI-DON	1.47e-4 ± 1.82e-5	1.42e-4 [1.37e-4, 1.47e-4]	2.22e-4
Burgers	RNO	1.50e-6 ± 1.40e-7	1.52e-6 [1.38e-6, 1.60e-6]	1.80e-6
Burgers	FNO_1	1.84e-6 ± 1.60e-7	1.81e-6 [1.73e-6, 1.98e-6]	2.30e-6
Burgers	FNO_5	7.17e-7 ± 1.14e-7	7.17e-7 [6.50e-7, 8.07e-7]	1.03e-6
CH	TI-DON	1.06e0 ± 7.76e-2	1.01e0 [9.98e-1, 1.17e0]	1.17e0
CH	RNO	9.34e-6 ± 1.59e-7	9.29e-6 [9.23e-6, 9.51e-6]	9.58e-6
CH	FNO_1	1.32e-7 ± 5.66e-8	1.65e-7 [6.26e-8, 1.85e-7]	1.99e-7
CH	FNO_5	5.93e-7 ± 1.64e-7	6.10e-7 [3.91e-7, 7.76e-7]	8.13e-7
CDR	TI-DON	1.95e-2 ± 3.14e-4	1.94e-2 [1.92e-2, 1.97e-2]	2.03e-2
CDR	RNO	1.09e-6 ± 3.16e-7	9.93e-7 [8.09e-7, 1.41e-6]	1.73e-6
CDR	FNO_1	8.37e-8 ± 1.46e-8	8.01e-8 [7.06e-8, 8.95e-8]	1.16e-7
CDR	FNO_5	5.85e-8 ± 1.21e-8	5.62e-8 [4.87e-8, 7.16e-8]	7.83e-8
KdV	TI-DON	2.31e-4 ± 7.56e-6	2.28e-4 [2.25e-4, 2.36e-4]	2.53e-4
KdV	RNO	8.85e-6 ± 1.67e-6	8.55e-6 [7.59e-6, 9.57e-6]	1.46e-5
KdV	FNO_1	1.28e-5 ± 2.45e-6	1.24e-5 [1.11e-5, 1.42e-5]	2.18e-5
KdV	FNO_5	6.38e-6 ± 7.76e-7	6.40e-6 [5.79e-6, 6.80e-6]	8.85e-6
KS	TI-DON	8.48e-4 ± 4.67e-6	8.47e-4 [8.45e-4, 8.51e-4]	8.60e-4
KS	RNO	4.33e-5 ± 1.44e-5	4.05e-5 [2.76e-5, 6.07e-5]	6.64e-5
KS	FNO_1	1.35e-5 ± 7.49e-6	1.05e-5 [7.29e-6, 1.99e-5]	3.53e-5
KS	FNO_5	6.79e-6 ± 4.05e-6	4.79e-6 [3.92e-6, 9.34e-6]	1.92e-5

Table 13: Open-loop forward-model endpoint MSE.

PDE	Model	Mean ± std	Median [q_{25}, q_{75}]	Maximum
AC	TI-DON	7.14e-4 ± 8.03e-4	4.36e-4 [1.89e-4, 9.22e-4]	4.20e-3
AC	RNO	3.55e-4 ± 5.38e-4	2.09e-4 [1.38e-4, 3.23e-4]	3.58e-3
AC	FNO_1	4.51e-5 ± 4.27e-5	3.14e-5 [1.59e-5, 6.07e-5]	1.86e-4
AC	FNO_5	9.17e-7 ± 1.26e-6	3.76e-7 [1.74e-7, 1.54e-6]	7.16e-6
Burgers	TI-DON	1.46e-3 ± 2.49e-3	3.89e-4 [2.02e-4, 1.62e-3]	1.34e-2
Burgers	RNO	1.09e-4 ± 2.50e-4	1.95e-5 [1.27e-5, 5.28e-5]	1.33e-3
Burgers	FNO_1	9.61e-5 ± 2.11e-4	2.45e-5 [1.61e-5, 6.11e-5]	1.31e-3
Burgers	FNO_5	3.94e-6 ± 6.15e-6	2.01e-6 [1.51e-6, 3.16e-6]	3.98e-5
CH	TI-DON	4.61e-6 ± 5.01e-6	3.21e-6 [1.85e-6, 5.10e-6]	2.34e-5
CH	RNO	7.83e-6 ± 9.23e-6	4.19e-6 [3.24e-6, 7.63e-6]	5.27e-5
CH	FNO_1	6.31e-6 ± 1.18e-5	1.24e-6 [3.23e-7, 5.40e-6]	5.22e-5
CH	FNO_5	2.71e-6 ± 2.28e-6	2.17e-6 [1.15e-6, 3.28e-6]	1.03e-5
CDR	TI-DON	8.82e-2 ± 1.15e-1	2.95e-2 [1.61e-2, 1.08e-1]	5.11e-1
CDR	RNO	1.06e-3 ± 8.29e-4	7.70e-4 [5.90e-4, 1.06e-3]	4.95e-3
CDR	FNO_1	5.43e-4 ± 1.36e-3	1.23e-5 [3.53e-6, 1.27e-4]	5.59e-3
CDR	FNO_5	5.28e-6 ± 1.23e-5	1.83e-7 [8.40e-8, 2.90e-6]	6.83e-5
KdV	TI-DON	3.81e-3 ± 1.12e-2	2.90e-4 [1.66e-4, 3.69e-3]	7.54e-2
KdV	RNO	2.90e-3 ± 1.37e-2	1.22e-4 [7.88e-5, 3.55e-4]	9.65e-2
KdV	FNO_1	5.16e-4 ± 1.28e-3	1.39e-4 [8.07e-5, 2.83e-4]	7.08e-3
KdV	FNO_5	5.61e-6 ± 2.05e-5	4.32e-7 [2.87e-7, 7.81e-7]	1.29e-4
KS	TI-DON	1.60e-1 ± 5.06e-1	1.77e-2 [7.70e-3, 4.79e-2]	3.07e0
KS	RNO	1.22e-1 ± 6.96e-1	5.59e-3 [3.24e-3, 9.93e-3]	4.93e0
KS	FNO_1	1.04e-1 ± 6.40e-1	6.49e-4 [3.71e-4, 5.91e-3]	4.53e0
KS	FNO_5	6.06e-4 ± 3.36e-3	4.57e-5 [1.86e-5, 9.40e-5]	2.38e-2

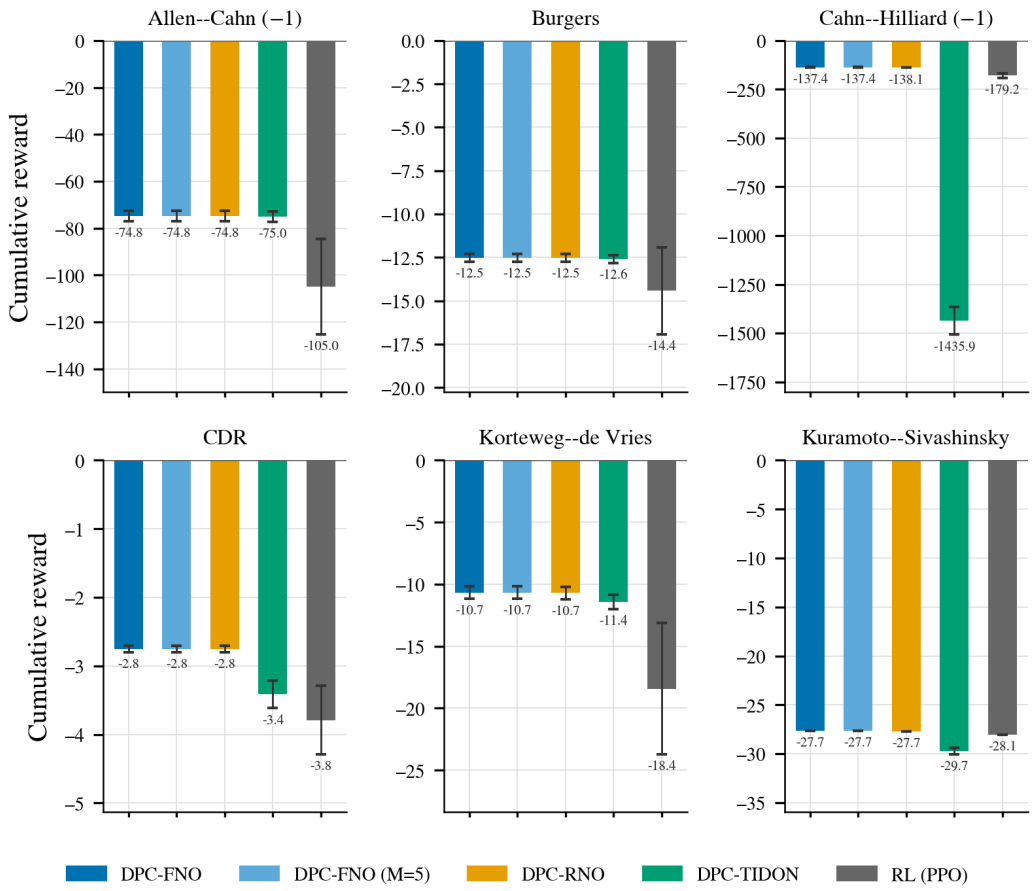

Figure 12: Closed-loop cumulative reward (mean ± std over 3 seeds) for each DPC surrogate model (FNO, FNO-MS, RNO, TIDON) versus the RL(PPO) baseline, across the six controlgym PDEs. Reward is $-N\,J(\pi;\boldsymbol{u}_0)$, the negative, unnormalized rollout cost (Eq. 1a), computed via controlgym's environment reward with identical $\boldsymbol{Q}$, $\boldsymbol{R}$ across all PDEs. Each panel uses an independent y-axis to accommodate differing reward scales across PDEs. Bar labels report the mean to one decimal place; higher (less negative) is better.

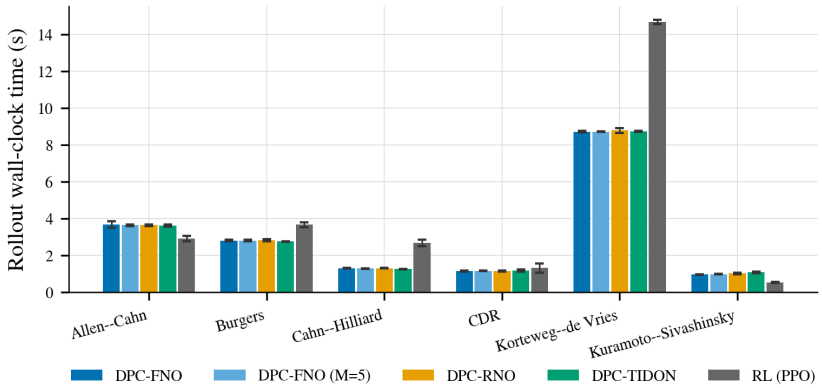

Figure 13: Plant roll-out wall-clock time (mean ± std over 3 seeds) for each DPC surrogate model versus the RL(PPO) baseline, across the six controlgym PDEs, on a shared y-axis

Representative closed-loop rollouts. Figures 14–19 use the same three-panel layout as the Cahn–Hilliard example in Figure 4.

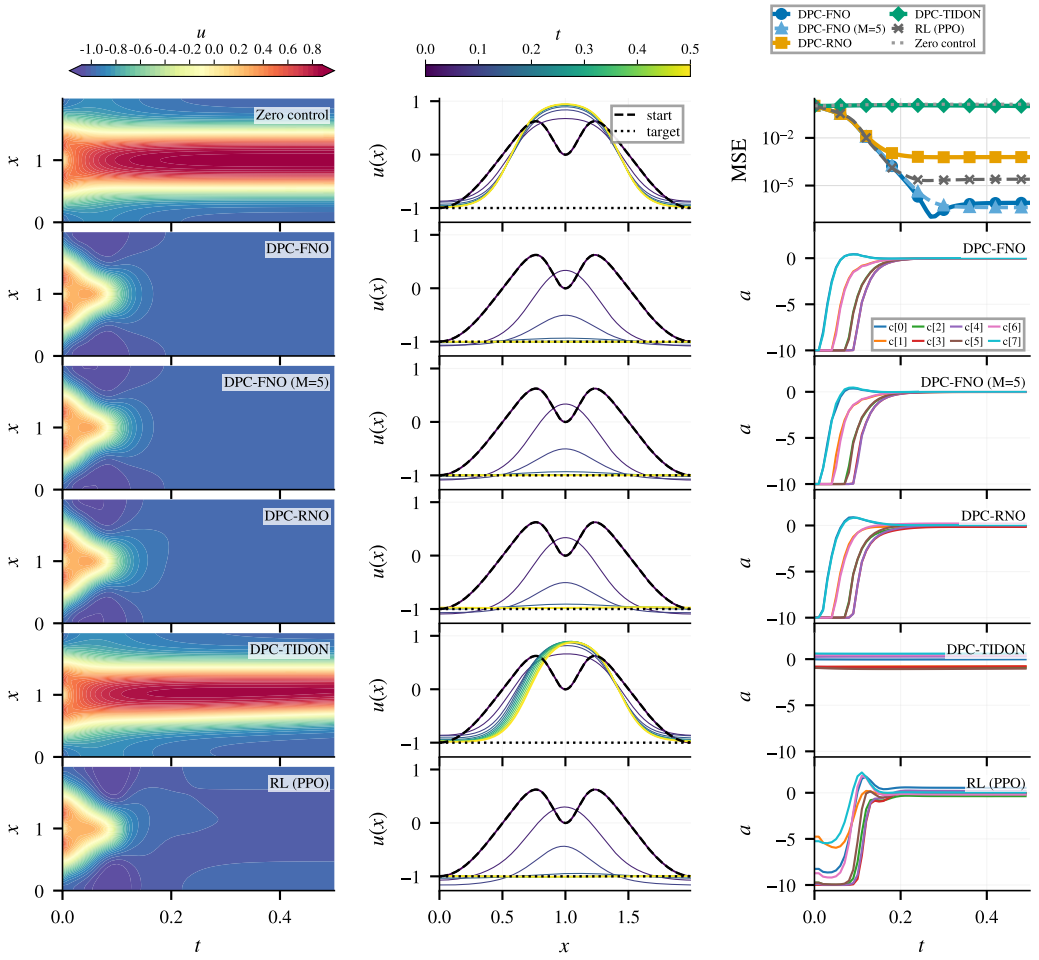

Figure 14: **Representative closed-loop rollout for Cahn–Hilliard.** The reference is the uniform state -1.

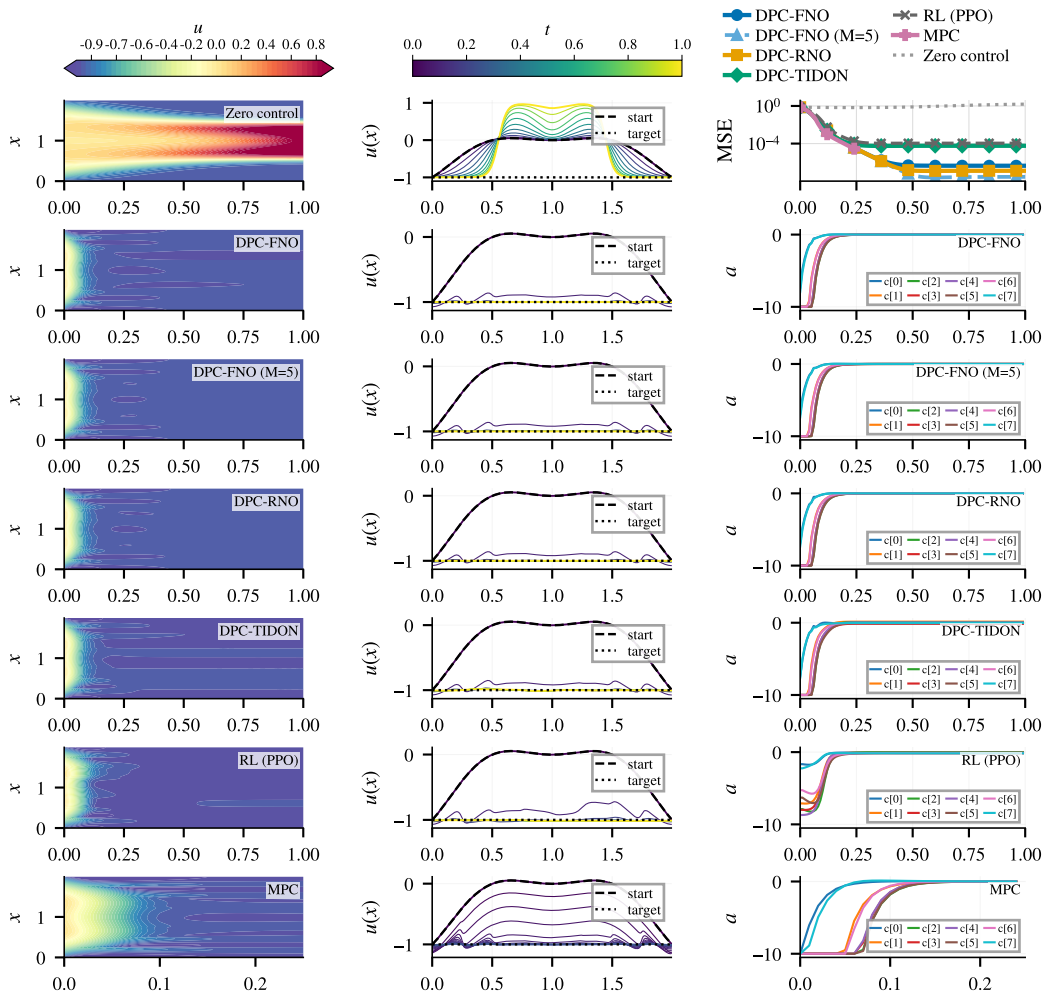

Figure 15: **Representative closed-loop rollout for Allen–Cahn.** The reference is the uniform state -1.

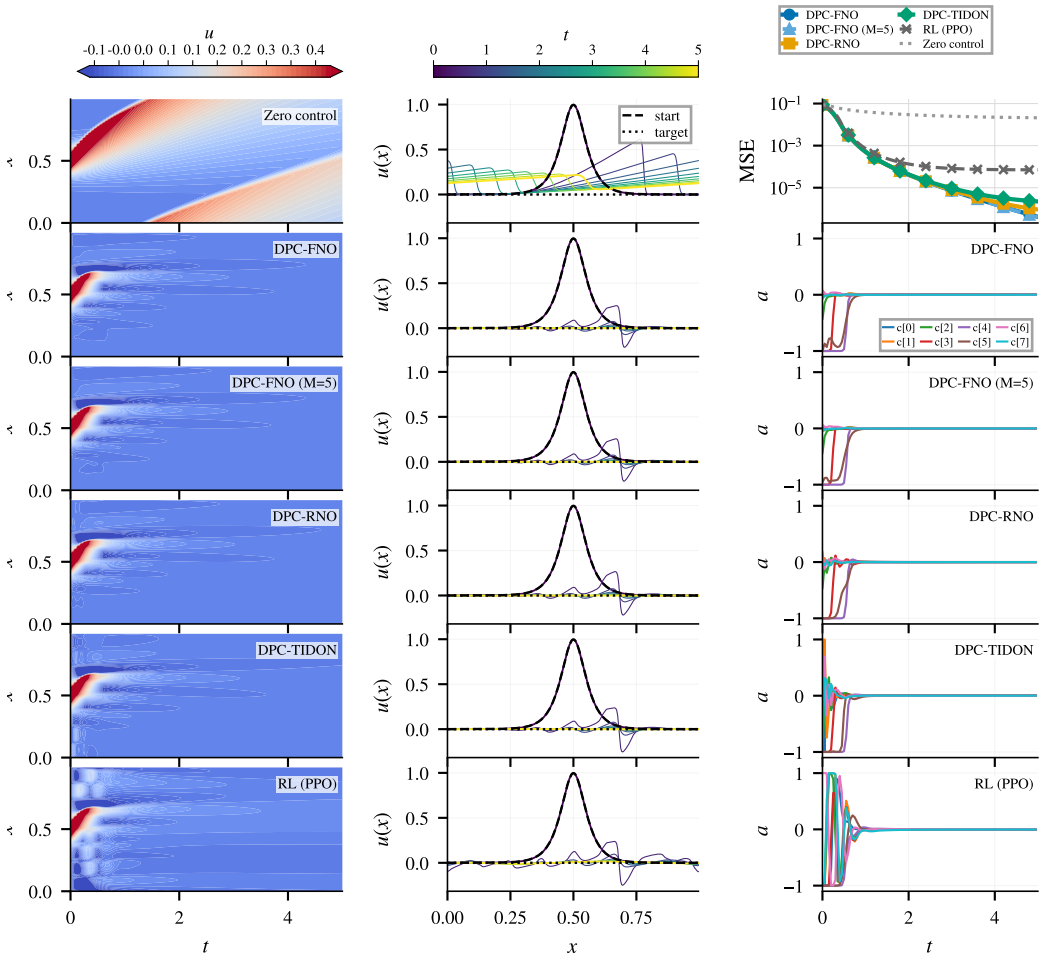

Figure 16: **Representative closed-loop rollout for Burgers.** The reference is zero.

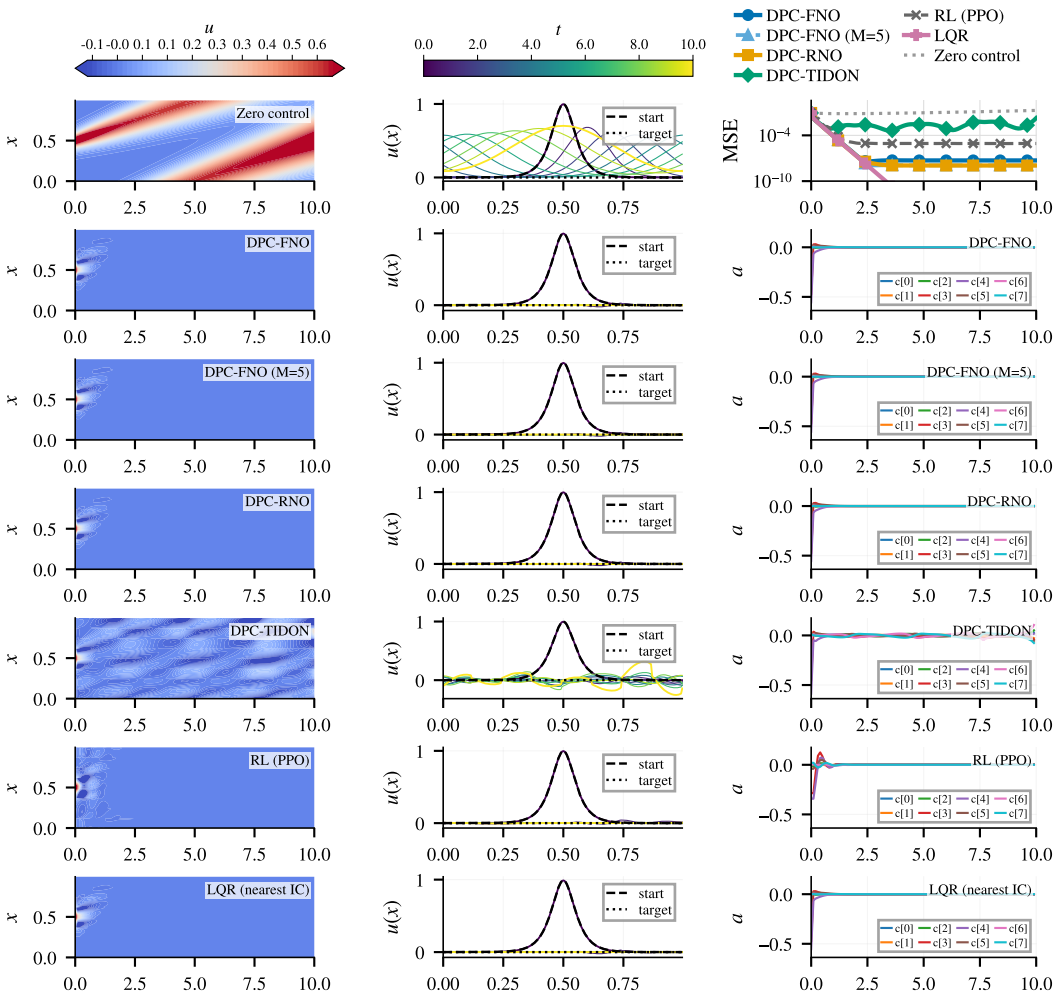

Figure 17: **Representative closed-loop rollout for convection–diffusion–reaction.** The reference is zero.

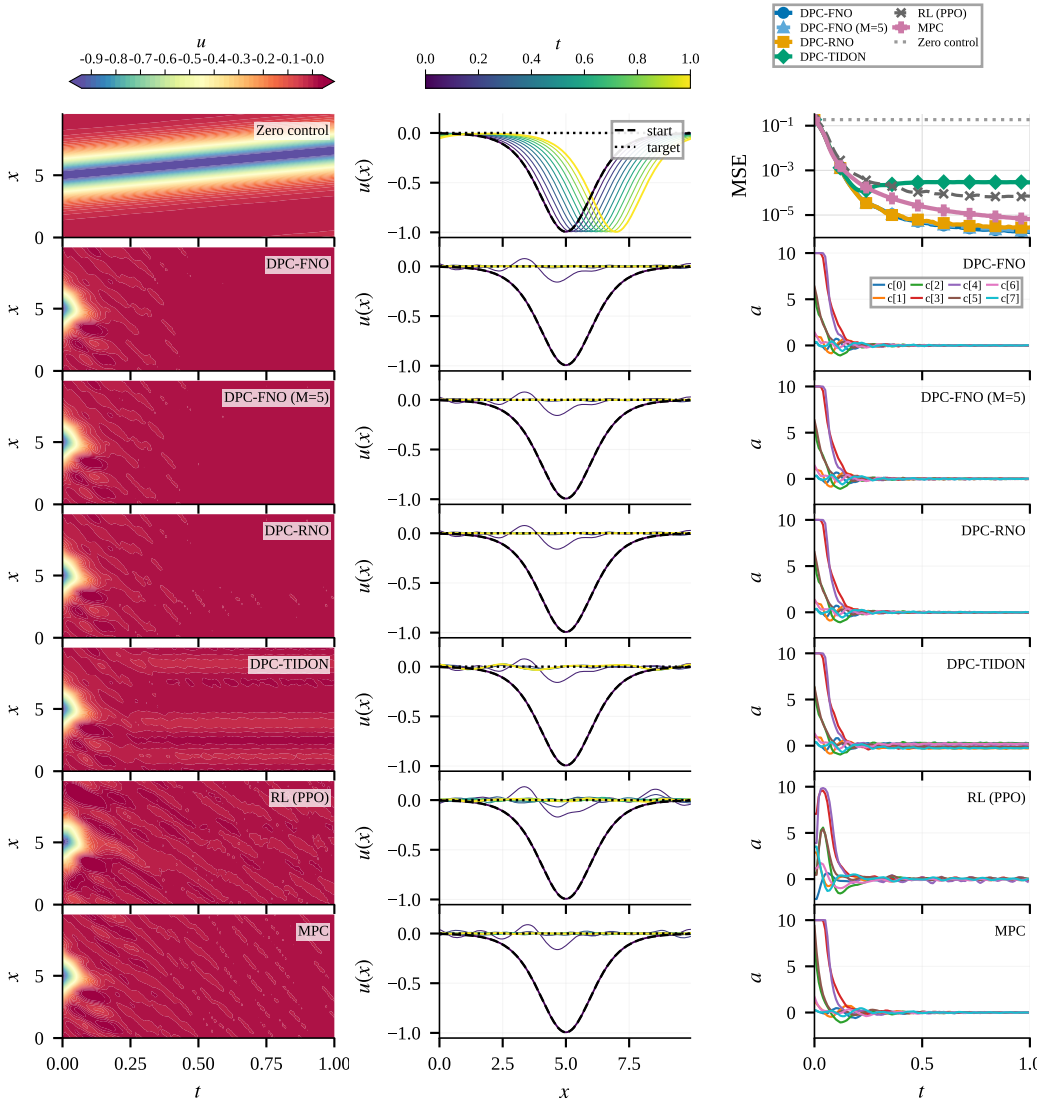

Figure 18: **Representative closed-loop rollout for Korteweg–de Vries.** The reference is zero.

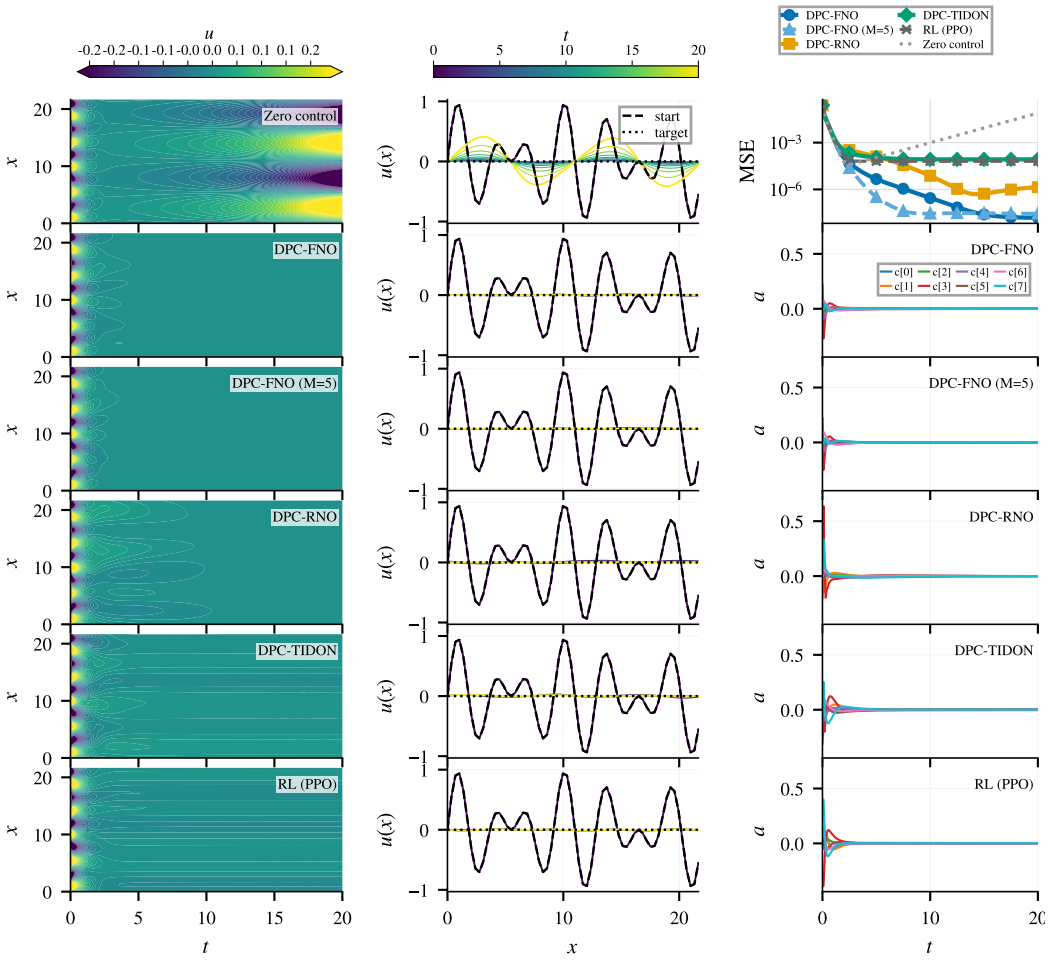

Figure 19: **Representative closed-loop rollout for Kuramoto–Sivashinsky.** The domain length is $L = 22$, and the reference is zero.