LEARNING TO CONTROL ON THE FLY: ONLINE META FLOW MATCHING ACROSS UNKNOWN DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

The control of complex physical systems is a fundamental challenge in science and engineering, requiring effective finite-horizon planning under unknown and varying dynamics. While deep learning provides a promising framework for optimizing control sequences, most learning-based approaches assume a fixed distribution of system dynamics. This assumption limits their ability to adapt efficiently when system dynamics are either unknown in advance or vary across environments. To address this limitation, we propose Online Meta Flow Matching (OMFM), a generative control framework that continuously adapts control generation from streaming interactions without requiring explicit knowledge of governing equations or dynamical parameters. For each encountered environment, OMFM encodes observed trajectories into a shared context that adapts both control generation and candidate evaluation to the underlying dynamics. The generator learns to produce control sequences from a Gibbs-weighted empirical distribution of observed trajectories, while the predictor estimates future state evolution for candidate evaluation. During deployment, multiple candidates are generated, evaluated by a learned dynamics predictor, and ranked by their predicted objectives without additional environment interaction. We evaluate OMFM on five diverse control tasks involving dynamical systems of varying complexity. The results indicate that OMFM consistently outperforms strong baselines, achieving 25.01% improvement in control performance and converging significantly faster during online adaptation. Our code is available at https://anonymous.4open.science/r/OMFM-473B.

1 INTRODUCTION

The control of complex physical systems remains a significant challenge in science and engineering. Achieving effective control requires anticipating system evolution and optimizing time-dependent inputs to guide system dynamics toward specific objectives. These control challenges are encountered in diverse applications, such as robotic manipulation (Chi et al., 2025; Gu et al., 2017), controlled nuclear fusion (Degrave et al., 2022), underwater systems (Zhang et al., 2022), and other physical dynamical systems (Holl et al., 2020a; Akhare et al., 2026). The nonlinear and high-dimensional nature of these systems further complicates the development of robust control strategies, necessitating the optimization of long-horizon control trajectories under complex conditions. In many cases, control is implemented through repeated, finite-duration experiments, episodes, or process runs, where observations and outcomes from completed batches inform subsequent operations (Srinivasan & Bonvin, 2007; Yoo et al., 2021).

Various control paradigms have been developed for complex dynamical systems. Classical optimal control and trajectory optimization generate control trajectories from known dynamics using numerical techniques such as direct collocation, sequential quadratic programming, and differential dynamic programming (Kelly, 2017; Malyuta et al., 2021; Mastalli et al., 2022). Model predictive control (MPC) enables closed-loop control by repeatedly solving finite-horizon optimization problems; however, this process can be computationally intensive (Hewing et al., 2020; Weinan et al., 2022), particularly for high-dimensional systems that require costly simulation or differentiation (Holl et al., 2020a; Wei et al., 2024). In contrast, learning-based approaches utilize offline trajectories to amortize the cost of prediction, planning, or policy optimization (Lillicrap et al., 2015; Chua et al., 2018). Recently, generative models have been employed to learn distributions over entire con-

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.

trol trajectories, enabling sampling of multiple candidate solutions through learned inference rather than iterative optimization (Janner et al., 2022; Wei et al., 2024; Chi et al., 2025; Hu et al., 2025).

Despite recent advances, most offline learning-based control methods assume that system dynamics encountered during deployment are adequately represented by a *fixed* training distribution. This assumption is often restrictive in real-world physical systems, where dynamics may be initially unknown or may vary across successive batches due to changing operating conditions, physical properties, and external disturbances (Nakhaei et al., 2024; Lin et al., 2024). While batch-to-batch and run-to-run control approaches utilize information from completed runs to enhance subsequent operations (Liu et al., 2018; Ye et al., 2022), these methods primarily address repeated processes rather than facilitating the transfer of control knowledge across heterogeneous dynamical environments. Consequently, controllers trained offline on fixed dynamics frequently generalize poorly to previously unseen environments, and collecting large datasets for retraining is often impractical (Nakhaei et al., 2024; Bhardwaj et al., 2024). These challenges highlight the need for online learning across sequentially encountered batches and dynamical systems, enabling the reuse of experience from prior environments to improve future control (Shi et al., 2021; Levy et al., 2025; Richards et al., 2023; Mei et al., 2025). The central challenge is to infer control-relevant information about current dynamics from limited interactions and to continually accumulate transferable knowledge across batches, since identical initial states can require different controls under different underlying dynamics (Richards et al., 2023; Bhardwaj et al., 2024; Sanghvi et al., 2024).

In this paper, we introduce Online Meta Flow Matching (OMFM), a generative online meta-learning framework for control under unknown and varying dynamics. OMFM learns transferable control knowledge from sequentially arriving trajectory data. Its Transition-Aware Context Encoder (TACE) extracts compact environment representations by preserving temporal transition information within trajectories and aggregating information across trajectories, and the resulting context conditions both the control generator and dynamics predictor. The generator learns from a Gibbs-weighted empirical distribution that emphasizes lower-cost control trajectories, while the predictor models future state evolution for candidate evaluation. At deployment, OMFM generates multiple candidate control sequences and selects the one with the lowest predicted control objective without additional environment interaction.

OMFM performs online learning across a sequence of training environments with bounded replay, enabling continuous accumulation of transferable control knowledge while retaining past experience. OMFM adapts to novel environments through trajectory-derived context, keeping its learned parameters fixed during control generation and candidate evaluation. After training, model parameters remain constant, and a single observed trajectory generates the environment representation for control generation and candidate ranking.

Our main contributions are as follows: (1) We introduce an online control framework in which controllers learn from sequentially encountered environments with limited memory and without access to privileged dynamics information. (2) We introduce OMFM, which leverages trajectory-derived context to adapt both control generation and candidate evaluation for systems with unknown dynamics. (3) We show that OMFM consistently reduces control costs by 11.70%–43.16% compared with the strongest baseline across five systems.

2 PRELIMINARIES

2.1 PROBLEM SETTING

We consider an **online learning** setting in which environments arrive sequentially and the **underlying dynamics parameters may change across environments**. Let $\mathcal{S} \subseteq \mathbb{R}^{d_s}$ and $\mathcal{U} \subseteq \mathbb{R}^{d_u}$ be the state and control spaces. Each environment e has an unobserved parameter $\xi_e \in \Xi$, where Ξ is the parameter space. Its dynamics follow $s_{t+1} = f_{\xi_e,t}(s_t, u_t)$, with $f_{\xi_e,t} : \mathcal{S} \times \mathcal{U} \to \mathcal{S}$, allowing time-dependent dynamics. We learn from the sequential training stream $\mathcal{E}_{\mathrm{tr}} = \{e_i^{\mathrm{tr}}\}_{i=1}^{N_{\mathrm{tr}}}$ and evaluate on $\mathcal{E}_{\mathrm{te}} = \{e_i^{\mathrm{te}}\}_{i=1}^{N_{\mathrm{te}}}$, as shown in Figure 1. Their parameter sets $\Xi_{\mathrm{tr}} = \{\xi_e : e \in \mathcal{E}_{\mathrm{tr}}\}$ and $\Xi_{\mathrm{te}} = \{\xi_e : e \in \mathcal{E}_{\mathrm{te}}\}$ satisfy $\Xi_{\mathrm{tr}} \cap \Xi_{\mathrm{te}} = \varnothing$. Within each split, i indexes environments and j trajectories. Training environment e_i^{tr} provides

$$\mathcal{D}_i = \{(\tau_{i,j}, g_{i,j}, J_{i,j})\}_{j=1}^{n_i}, \quad \tau_{i,j} = \{(s_{i,j,t}, u_{i,j,t}, s_{i,j,t+1})\}_{t=0}^{H_{i,j}-1}, \tag{1}$$

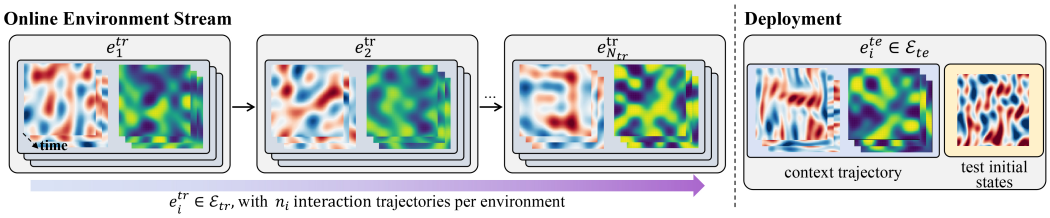


Figure 1: Sequential training across environments and deployment in unseen environments using a single observed trajectory for adaptation.

where $H_{i,j}$ denotes the horizon, and $g_{i,j} \in \mathcal{G}$ specifies the task-level control condition, including the initial state, desired reference, and horizon, with $\mathcal{G}$ denoting the task condition space. Then we define $\xi_i := \xi_{e_i^{\text{tr}}}$, $X_{i,j} := (s_{i,j,t})_{t=0}^{H_{i,j}}$, and $U_{i,j} := (u_{i,j,t})_{t=0}^{H_{i,j}-1} \in \mathcal{U}^{H_{i,j}}$. The cost is given by $J_{i,j} = J(X_{i,j}, U_{i,j}; g_{i,j}) = J_{\xi_i}(U_{i,j}; g_{i,j}) \in \mathbb{R}$, where J_{ξ_i} evaluates the trajectory cost J under the true dynamics and initial state specified by $g_{i,j}$. Both n_i and $H_{i,j}$ may vary. For methods that exploit environment context, a context set for training environment e_i^{tr} is formed by sampling n_c trajectories from $\mathcal{D}_i$:

$$\mathcal{C}_{e_i^{\text{tr}}} = \{\tau_{i,j} : j \in \mathcal{K}_i\}, \quad \mathcal{K}_i \subseteq \{1, \ldots, n_i\}, \quad 1 \le |\mathcal{K}_i| = n_c < n_i. \tag{2}$$

The complementary training pool is

$$\mathcal{A}_i = \{1, \ldots, n_i\} \setminus \mathcal{K}_i, \quad \mathcal{D}_i^{\text{train}} = \{(\tau_{i,j}, g_{i,j}, J_{i,j}) : j \in \mathcal{A}_i\}. \tag{3}$$

Context-based methods map $\mathcal{C}_{e_i^{\text{tr}}}$ to an environment representation $c_i \in \mathbb{R}^{d_c}$, shared across control instances within the environment. This representation conditions the control model or dynamics predictor. In contrast, context-free methods learn directly from trajectories and task conditions without environment-specific representations. In experiments, each previously unseen test environment provides a single trajectory for context-based adaptation; method-specific evaluation procedures are given in Appendix D.1. The objective is to achieve low-cost control in these environments, while maintaining performance throughout sequential training with bounded replay.

2.2 Online Amortized Meta-Learning

Online amortized meta-learning learns a shared context-dependent adaptation mechanism across sequentially arriving environments. Let Θ_i denote the model parameters after processing the first i training environments, and let Θ denote a generic parameter variable during optimization. At round i, learning uses only the current data $\mathcal{D}_i$ and past data retained in the replay buffer. The replay buffer $\mathcal{R}_{i-1} \subseteq \{\mathcal{D}_{i'} : 1 \le i' < i\}$ stores trajectory data from at most B past environments, with $|\mathcal{R}_{i-1}| \le B$. Let $\mathcal{B}_i$ index the environments sampled from this buffer. The objective $\ell_i(\Theta)$ averages the losses over environments indexed by $\{i\} \cup \mathcal{B}_i$, in expectation over sampled context sets and their complementary training pools. An illustrative gradient step is:

$$\Theta_i = \Theta_{i-1} - \eta_i \, \nabla_\Theta \ell_i(\Theta)|_{\Theta=\Theta_{i-1}}, \quad \eta_i > 0, \tag{4}$$

where η_i is the learning rate at round i. The implementation applies a task-specific number of optimization steps for each data arrival. Adaptation is achieved through c_i, eliminating the need for task-specific inner optimization or parameter updates during testing. Section 3.2 provides detailed specifications of the loss functions and replay construction.

2.3 Conditional Flow Matching

For conditioning information y, Conditional Flow Matching (CFM) learns a conditional vector field for transporting a reference distribution $p_0(\cdot \mid y)$ to a target distribution $p_1(\cdot \mid y)$ (Lipman et al., 2023). Draw $(W_0, W_1) \sim \varpi(\cdot, \cdot \mid y)$, a coupling with these marginals, and independently draw $\alpha \sim \text{Unif}[0, 1]$. The linear path $W_\alpha = (1-\alpha)W_0 + \alpha W_1$ has velocity $V_\alpha = W_1 - W_0$. A vector field parameterized by θ minimizes

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}\left[\|v_\theta(W_\alpha, \alpha \mid y) - V_\alpha\|_2^2\right]. \tag{5}$$

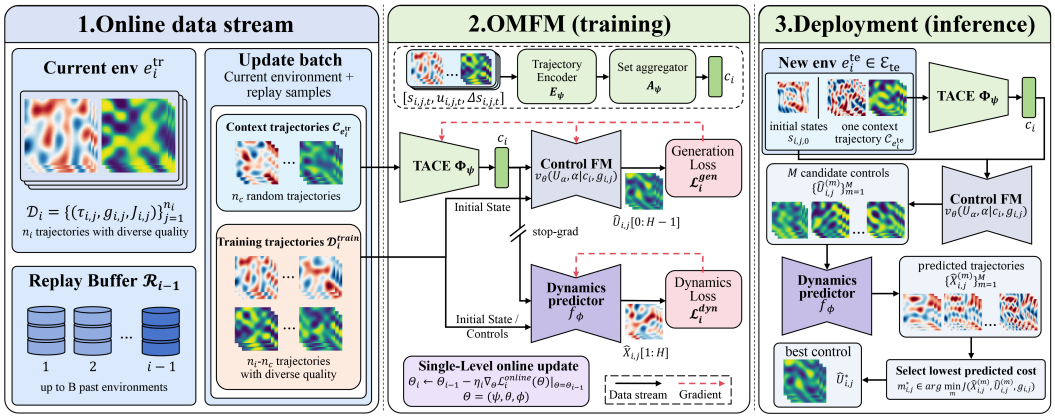


Figure 2: Overview of OMFM. Online learning accumulates knowledge across environments, while context-based conditioning enables control generation and predictor-assisted selection in unseen environments without parameter updates.

Generation draws $W_0 \sim p_0(\cdot \mid y)$ and integrates $dW_\alpha/d\alpha = v_\theta(W_\alpha, \alpha \mid y)$ from $\alpha = 0$ to 1. The learned field turns reference samples into control candidates; Sec. 3.2 specifies the conditioning, and Appendix B gives the regression interpretation.

3 Methodology

The primary modeling challenge is to condition a control generator on an environment with unobserved dynamics. During training on $\mathcal{E}_{\text{tr}}$, trajectories are encoded using TACE, and the resulting representation is used to condition a control-space flow-matching (FM) model (Lipman et al., 2023). The encoder, generator, and dynamics predictor are updated online with the current training environment and bounded replay. For an unseen environment in $\mathcal{E}_{\text{te}}$, all learned parameters remain fixed. The context representation then conditions the flow, which proposes multiple controls for model-based selection.

3.1 Transition-Aware Environment Representation

An environment index identifies an environment but says nothing about its unknown transition law. We instead treat each context trajectory in Eq. 2 as one observation of the environment's behavior and define the empirical measure:

$$\widehat{\rho}_i = \frac{1}{n_c} \sum_{j \in \mathcal{K}_i} \delta_{\tau_{i,j}} \in \mathcal{P}(\mathcal{T}), \tag{6}$$

where $\mathcal{T}$ is the trajectory space, $\mathcal{P}(\mathcal{T})$ is the set of probability measures on $\mathcal{T}$, and δ_τ denotes a point mass at trajectory τ. This empirical measure yields a distributional interpretation of the sampled context, and the encoder receives the finite trajectory set $\mathcal{C}_{e_i^{\text{tr}}}$. Employing trajectories instead of isolated transition tuples preserves the temporal response to a control sequence. We assume that environments with the same context-trajectory law induce the same task-relevant control law for almost every control condition under the task-condition distribution.

A shared vector field adapts to each environment through its trajectory-derived context representation. Let Φ_ψ denote TACE with parameters ψ. This function maps sampled context trajectories to an environment representation $c_i \in \mathbb{R}^{d_c}$, which specifies the environment-specific control flow:

$$\mathcal{C}_{e_i^{\text{tr}}} \longmapsto c_i = \Phi_\psi(\mathcal{C}_{e_i^{\text{tr}}}) \longmapsto v_\theta(\cdot \mid c_i, g_{i,j}). \tag{7}$$

This provides a sample-based realization of measure-conditioned amortization: the representation of the empirical context in Eq. 6 conditions a shared family of control vector fields (Atanackovic et al., 2025).

Pooling all transitions directly discards the temporal order in which the environment responds to a control sequence. Our TACE method instead preserves temporal structure within each context trajectory and applies permutation-invariant aggregation only across different trajectories. For a context trajectory $\tau_{i,j}$ with $j \in \mathcal{K}_i$, the lowercase vector $\mathbf{x}_{i,j,t}$ denotes the single-step transition feature at time t, whereas the uppercase $X_{i,j}$ represents the complete state sequence:

$$\mathbf{x}_{i,j,t} = \big[s_{i,j,t},\, u_{i,j,t},\, \Delta s_{i,j,t}\big], \quad \Delta s_{i,j,t} = s_{i,j,t+1} - s_{i,j,t}, \quad \mathbf{x}_{i,j,t} \in \mathbb{R}^{2d_s+d_u}. \tag{8}$$

TACE first encodes each context trajectory and then aggregates the trajectory representations into a shared environment context:

$$c_i = A_\psi\Big(\Big\{E_\psi\Big((\mathbf{x}_{i,j,t})_{t=0}^{H_{i,j}-1}\Big) : j \in \mathcal{K}_i\Big\}\Big) \in \mathbb{R}^{d_c}, \tag{9}$$

where E_ψ encodes temporal dependencies within each trajectory, while A_ψ aggregates information across trajectories in a permutation-invariant manner (Zaheer et al., 2017). The parameter set ψ encompasses all trainable parameters for these modules. Pooling concatenates mean and maximum statistics first over time and then across trajectories. This transition-based representation facilitates latent task inference (Rakelly et al., 2019): c_i summarizes environment behavior, rather than explicitly estimating ξ_i, and conditions both the control generator and the dynamics predictor. The context c_i is shared across all control instances within the environment, whereas $g_{i,j}$ specifies the individual control condition.

3.2 Context-Conditioned Control Learning

For environment e_i^{tr} and control condition $g_{i,j}$, let $q_{\xi_i}(\cdot \mid g_{i,j})$ denote a proposal distribution over control sequences. We represent low-cost behavior by the Gibbs-tilted control measure:

$$\begin{aligned} \nu^{\beta}_{\xi_i}(dU \mid g_{i,j}) &= \frac{\exp[-\beta J_{\xi_i}(U; g_{i,j})]}{Z^{\beta}_{\xi_i}(g_{i,j})} q_{\xi_i}(dU \mid g_{i,j}), \\ Z^{\beta}_{\xi_i}(g_{i,j}) &= \int \exp[-\beta J_{\xi_i}(U; g_{i,j})] q_{\xi_i}(dU \mid g_{i,j}), \end{aligned} \tag{10}$$

where $0 < Z^{\beta}_{\xi_i}(g_{i,j}) < \infty$ and $\beta \geq 0$ controls the concentration on low-cost controls, with $\beta = 0$ recovering the proposal distribution. We apply the CFM construction in Sec. 2.3 to control sequences, conditioning the shared vector field on $y = (c_i, g_{i,j})$. At the population level, environment e_i^{tr} has reference distribution $\gamma(\cdot \mid g_{i,j})$ and desired target distribution $\nu^{\beta}_{\xi_i}(\cdot \mid g_{i,j})$. Draw independent endpoints $U_0 \sim \gamma(\cdot \mid g_{i,j})$ and $U_1 \sim \nu^{\beta}_{\xi_i}(\cdot \mid g_{i,j})$, and independently draw $\alpha \sim \mathrm{Unif}[0,1]$. The interpolation and target velocity are $U_\alpha = (1-\alpha)U_0 + \alpha U_1$ and $V_\alpha = U_1 - U_0$. The corresponding CFM objective is

$$\mathcal{L}_{\mathrm{CFM}}(\theta, \psi; \mathcal{C}_{e_i^{\mathrm{tr}}}, g_{i,j}) = \mathbb{E}_{U_0,U_1,\alpha}\Big[\big\|v_\theta(U_\alpha, \alpha \mid c_i, g_{i,j}) - V_\alpha\big\|_2^2\Big]. \tag{11}$$

For CFM pooled across environments, the unrestricted context-conditioned population minimizer is $v^\star(u, \alpha \mid c_i, g_{i,j}) = \mathbb{E}[V_\alpha \mid U_\alpha = u, \alpha, c_i, g_{i,j}]$, where $u \in \mathbb{R}^{H_{i,j}d_u}$ denotes a possible interpolated control sequence. The representation c_i is extracted from the context information $\hat{\rho}_i$ to characterize the underlying environment. The conditional vector field is the L^2 projection of endpoint velocities onto the information in $(U_\alpha, \alpha, c_i, g_{i,j})$. When c_i preserves the environment-conditioned endpoint coupling, this field recovers the corresponding environment-conditioned population velocity. The generated variable is the complete control sequence U, whose dimension $H_{i,j}d_u$ is independent of the state dimension. The finite training pool comprises controls of varying quality. All $n_i - n_c$ non-context trajectories are utilized, with each assigned a Gibbs weight according to its observed cost:

$$\widetilde{w}^{\beta}_{i,j} = \frac{\exp[-\beta(J_{i,j} - J^{\mathrm{train}}_{i,\min})]}{\sum_{j' \in \mathcal{A}_i} \exp[-\beta(J_{i,j'} - J^{\mathrm{train}}_{i,\min})]}, \quad J^{\mathrm{train}}_{i,\min} = \min_{j' \in \mathcal{A}_i} J_{i,j'}, \quad j \in \mathcal{A}_i. \tag{12}$$

The shift by $J^{\mathrm{train}}_{i,\min}$ preserves the normalized weights and enhances numerical stability. Each non-context pair $(g_{i,j}, U_{i,j})$ remains within the empirical training distribution, with lower-cost pairs assigned greater weight. Across the observed conditions, these weights define a Gibbs-weighted empirical distribution of condition–control pairs, providing the finite-data training surrogate for the conditional target in Eq. 10. During flow optimization, we hold observed controls, costs, and weights constant.

Online learning across training environments. Let f_ϕ denote the dynamics predictor with parameters ϕ, and collect all trainable parameters as $\Theta = (\psi, \theta, \phi)$. For environment e_i^{tr}, the generator loss $\mathcal{L}_i^{\mathrm{gen}}(\psi, \theta)$ is the empirical version of Eq. 11, using the Gibbs weights in Eq. 12. The dynamics predictor minimizes:

$$\mathcal{L}_i^{\mathrm{dyn}}(\phi) = \frac{1}{|\mathcal{A}_i|} \sum_{j \in \mathcal{A}_i} \mathcal{L}_{\mathrm{pred}}(f_\phi; X_{i,j}, U_{i,j}, \mathrm{sg}[c_i]) \,, \tag{13}$$

where sg denotes stop-gradient, keeping the encoder representation driven by the control-generation loss while the prediction loss trains the dynamics predictor. $\mathcal{L}_{\mathrm{pred}}$ combines a one-step prediction loss with a multi-horizon prediction loss obtained by recursively applying f_ϕ. This uniform average uses the same non-context trajectories as the generator, without cost weighting. The dynamics loss updates only ϕ, while the generation loss learns (ψ, θ).

Based on the current and replayed environments defined in Sec. 2.2, we construct one context representation $c_{i'} = \Phi_\psi(\mathcal{C}_{e_{i'}^{\mathrm{tr}}})$ for each environment $i' \in \{i\} \cup \mathcal{B}_i$. Each $c_{i'}$ is derived exclusively from the context trajectories of its respective environment and is shared across its non-context training trajectories. The online objective is then optimized as follows:

$$\mathcal{L}_i^{\mathrm{online}}(\Theta) = \frac{1}{1 + |\mathcal{B}_i|} \sum_{i' \in \{i\} \cup \mathcal{B}_i} \Big[\mathcal{L}_{i'}^{\mathrm{gen}}(\psi, \theta) + \lambda_{\mathrm{dyn}} \mathcal{L}_{i'}^{\mathrm{dyn}}(\phi) \Big] \,. \tag{14}$$

Here, λ_{dyn} weights the dynamics loss. For sampled context sets, this objective is a realization of ℓ_i in Eq. 4. We perform a small number of updates per arriving environment.

3.3 Control in Unseen Test Environments

For an unseen environment e_i^{te}, let $\mathcal{C}_{e_i^{\mathrm{te}}}$ denote its single observed context trajectory and encode it as $c_i = \Phi_\psi(\mathcal{C}_{e_i^{\mathrm{te}}})$. Given a control condition $g_{i,j}$, the flow induces a distribution $\pi_\theta(\cdot \mid c_i, g_{i,j})$ over complete control sequences. Integrating the learned vector field over flow time α with independent reference samples yields M candidates:

$$\widehat{U}_{i,j}^{(m)} \sim \pi_\theta(\cdot \mid c_i, g_{i,j}), \qquad m = 1, \ldots, M. \tag{15}$$

For each candidate, the dynamics predictor recursively estimates the state sequence from the prescribed initial state $\widehat{s}_{i,j,0}^{(m)} = s_0(g_{i,j})$:

$$\widehat{s}_{i,j,t+1}^{(m)} = f_\phi\Big(\widehat{s}_{i,j,t}^{(m)}, \widehat{u}_{i,j,t}^{(m)}, c_i\Big) \,, \qquad t = 0, \ldots, H_{i,j} - 1, \tag{16}$$

where $\widehat{u}_{i,j,t}^{(m)}$ is the t-th control in $\widehat{U}_{i,j}^{(m)}$. The resulting predicted trajectory is $\widehat{X}_{i,j}^{(m)} = (\widehat{s}_{i,j,t}^{(m)})_{t=0}^{H_{i,j}}$. Following model-based candidate evaluation (Nagabandi et al., 2018b; Chua et al., 2018; Argenson & Dulac-Arnold, 2021), we select the control with the lowest predicted cost:

$$m_{i,j}^* \in \operatorname*{arg\,min}_{1 \le m \le M} J\Big(\widehat{X}_{i,j}^{(m)}, \widehat{U}_{i,j}^{(m)}; g_{i,j}\Big) \,, \qquad \widehat{U}_{i,j}^* = \widehat{U}_{i,j}^{(m_{i,j}^*)}. \tag{17}$$

We use $M = 16$ in the experiments. All candidates are evaluated using the learned predictor, and only $\widehat{U}_{i,j}^*$ is executed open-loop over the prescribed horizon. Increasing M does not require additional physical interactions for selection. The complete training and deployment procedure is provided in Algorithm 1 in Appendix B.4.

4 Experiments

We evaluate OMFM on five dynamical control systems covering nonlinear low-dimensional dynamics and high-dimensional spatiotemporal processes: CartPole (Barto et al., 1983), Pendulum (Duan et al., 2016), the 1D Kuramoto–Sivashinsky (KS1D) equation (Bucci et al., 2019; Zeng & Graham, 2021), the 2D Kolmogorov Flow (K2D) (Kochkov et al., 2021; Li et al., 2024b), and a 2D smoke indirect control system (Smoke2D) (Wei et al., 2025). We consider multiple control settings for KS1D and Smoke2D, covering diverse dynamics, state/control dimensions, and objectives. The underlying dynamics vary across environments and are unavailable to the controller. Test environments include distribution shifts beyond training, with only one observed trajectory provided for environment adaptation. Detailed system configurations are given in Appendix C.

Table 1: Control performance in seven settings; improvement is relative to each column's best baseline. Mean J is reported; lower is better within each setting. Best results are shown in bold and second-best results are underlined. – denotes an inapplicable setting or non-convergent result.

Method	Pendulum	CartPole	KS1D		K2D	Smoke2D	
			Dissipation	Spectrum		Boundary	Domain
PID	0.1870	<u>12.2752</u>	–	–	–	–	–
Open-loop MPC	0.1875	12.6213	1.1697	0.0760	0.6127	–	–
M^{3}PC	0.2918	20.1344	1.6427	0.1627	0.6483	0.4268	0.5310
Sequence BC	0.1985	19.5605	0.5085	0.0684	0.2527	0.2950	0.2089
SAC-offline	0.2320	26.4381	0.4765	0.1462	0.5181	0.4253	–
SAC-surrogate	0.1783	13.3149	<u>0.1807</u>	0.0648	0.7364	0.4395	0.2046
BPPO	0.3223	20.4679	0.9126	0.1227	0.5239	0.5618	0.5001
FTML-Policy	<u>0.1547</u>	14.7674	0.2834	<u>0.0570</u>	<u>0.2309</u>	<u>0.2438</u>	<u>0.1601</u>
OFM	0.3310	18.9192	0.8686	0.0979	0.4484	0.3002	0.2889
FMPhyCon	0.2321	15.4553	0.2822	0.0786	0.3986	0.4605	0.3239
OMFM (Ours)	**0.1366**	**10.3238**	**0.1282**	**0.0324**	**0.1881**	**0.1915**	**0.1036**
Improvement	**11.70%**	**15.90%**	**29.05%**	**43.16%**	**18.54%**	**21.45%**	**35.29%**

4.1 Training Setup and Baselines

Training and evaluation. For each system, we construct a randomly ordered stream of training environments. Upon each arrival, models are optimized on the current environment and on samples from a past-only replay buffer containing the most recent 256 environments. All learned models are randomly initialized and trained with Adam using only streamed trajectories, without access to environment parameters or simulator calls. For OMFM, we set $\beta = 10$, $\lambda_{\text{dyn}} = 1$, and use 4 context trajectories during training. Evaluation uses fixed unseen in-distribution (ID) and out-of-distribution (OOD) environments. Each provides one trajectory for adaptation and multiple initial states for control evaluation. For each initial state, a control sequence is selected before a single ground-truth rollout, and the resulting objective J is reported as the primary metric. Further implementation details are provided in Appendix D.2.

Baselines. We compare OMFM with representative baselines under the same environment stream and evaluation protocol. Classical and model-based baselines include Proportional-Integral-Derivative (PID) control (Kumar & Ramadge, 2021), learned-dynamics MPC with cross-entropy optimization (Chua et al., 2018), and M^{3}PC, which performs test-time predictive control with a masked trajectory model trained on the same environment stream (Wen et al., 2025). Policy-learning baselines include sequence behavior cloning (BC) (Zhao et al., 2023), offline Soft Actor–Critic (SAC) (Haarnoja et al., 2018), a surrogate-augmented SAC variant, and Behavior Proximal Policy Optimization (BPPO) (Zhuang et al., 2023). We further evaluate a Follow-the-Meta-Leader (FTML) policy variant, FTML-Policy (Finn et al., 2019), as an online meta-learning baseline, using its learned initialization at test time. For generative control, we compare with our Online Fine-Tuned Flow Matching (OFM) baseline and our flow-matching variant of DiffPhyCon (FMPhyCon) (Wei et al., 2024). Additional details are provided in Appendix D.1.

Table 2: Robustness Evaluation on KS1D.

Method	Partial Observation	Measurement Noise	Control Noise
Sequence BC	0.5447	0.5109	0.5460
SAC-surrogate	0.5443	<u>0.1822</u>	<u>0.2093</u>
FTML-Policy	<u>0.3593</u>	0.2847	0.3129
FMPhyCon	0.3982	0.2905	0.3146
OMFM (Ours)	**0.1785**	**0.1293**	**0.1524**
Improvement	**50.32%**	**29.03%**	**27.19%**

Table 3: Ablation Results of OMFM on Pendulum Dataset.

Variant	$J \downarrow$	ΔJ
Full	**0.1366**	–
w/o Gibbs	0.1402	+2.64%
w/o Replay	0.1424	+4.25%
w/o Rank	0.1651	+20.86%
Joint FM	0.1693	+23.94%
w/o Encoder	0.1804	+32.06%

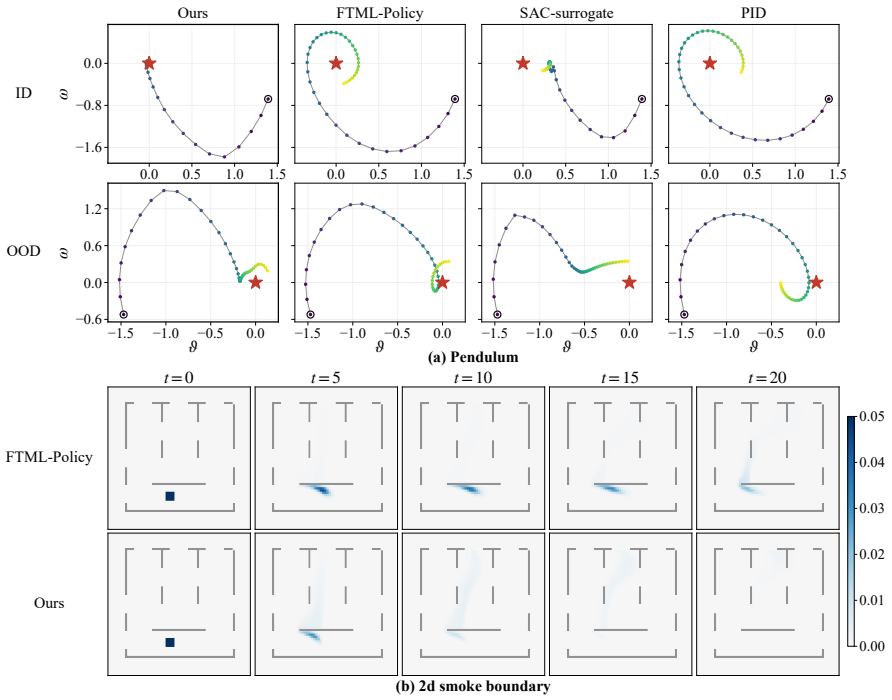


Figure 3: Representative rollouts illustrate how control performance translates into state evolution: OMFM avoids pronounced overshoot in the ID Pendulum example and leaves less residual smoke at intermediate times than FTML-Policy.

4.2 Main Results

Control performance. Table 1 compares OMFM with baseline methods across five dynamical systems and seven control settings. OMFM achieves the lowest mean objective values across all seven settings, improving J by 11.70% to 43.16% relative to the best-performing baseline. These gains appear across objectives in KS1D and actuation settings in Smoke2D, indicating robust advantages across diverse control configurations. The largest improvements are observed in KS1D spectral tracking and Smoke2D domain control, underscoring OMFM's effectiveness in shaping complex distributed dynamics. Figure 3 illustrates transient behaviors, showing that in the ID Pendulum example, OMFM reaches the target with less overshoot than FTML-Policy and PID. In Smoke2D boundary control, OMFM leaves less residual smoke at the displayed intermediate times than FTML-Policy. These results demonstrate that improvements in the objective function correspond to improved transient state trajectories. Additional ID/OOD comparisons and visualization results are provided in Appendices D.3 and D.4.

Robustness. Table 2 evaluates robustness under partial observation, measurement noise, and control noise. OMFM consistently outperforms the best competing baseline, reducing J by 50.32%, 29.03%, and 27.19% in the three settings, respectively. These results show that the performance gains persist under the tested KS1D observation and actuation perturbations.

4.3 Model Analysis

Online learning and generalization. Figure 4(a) evaluates control performance on a fixed test set as new training environments are introduced. OMFM improves rapidly on Pendulum and KS1D, whereas K2D continues to progress with additional experience. These results illustrate rapid control improvement and cumulative knowledge acquisition from the online training stream. Because test-time parameters remain constant, the observed gains reflect transferable learning across environments. Figure 4(b) demonstrates that OMFM sustains a larger low-J region than FTML-Policy within the K2D parameter space, including regions outside the training range. This finding supports generalization across the evaluated parameter variations.

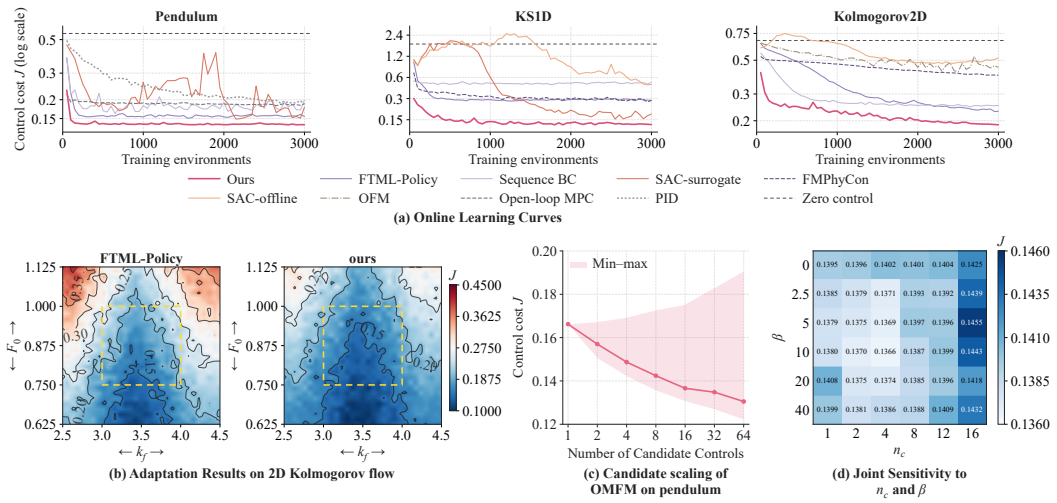


Figure 4: Evaluation of OMFM under different settings. (a) Online learning performance evaluated on a fixed test set throughout training. (b) Parameter-space evaluation on K2D. (c) Candidate scaling during inference. (d) Joint sensitivity to context size n_c and Gibbs weighting β.

Scaling and sensitivity analysis. Figure 4(c) examines test-time candidate scaling. Increasing the number of generated candidates gradually reduces the control objective, showing that sampling more controls from the learned distribution creates further opportunities for predictor-assisted candidate selection. The dynamics predictor ranks these candidates and selects trajectories with lower predicted objectives, demonstrating its ability to leverage expanded candidate diversity to improve control performance beyond a single generated sequence. This improvement requires only additional inference computation without model updates or physical interactions.

Figure 4(d) investigates the joint sensitivity to context size n_c and Gibbs weighting β. The results identify an effective allocation between environment context and control-generation supervision. Small contexts retain more trajectories for flow-matching training, while larger contexts supply additional observations for inferring environment behavior. Under the fixed trajectory budget, increasing context size allocates more trajectories to environment encoding and fewer to control-generation supervision. The default context size offers a practical balance between these effects. The lowest evaluated cost occurs at $n_c = 4$ and $\beta = 10$, supporting moderate cost weighting together with compact trajectory context.

Component ablations. Table 3 analyzes the contribution of each component. Removing the encoder increases J by 32.06%, demonstrating the importance of context-based environment representation for adapting to varying dynamics. Eliminating candidate ranking results in a 20.86% performance degradation, indicating that the predictor effectively identifies better trajectories among generated candidates. The Joint FM variant, which directly couples control generation and dynamics prediction, increases J by 23.94%, supporting the separation of control generation from predictive evaluation in the full model under the same Pendulum control setting. Removing Gibbs weighting and replay also degrades performance, supporting cost-weighted training and experience reuse as complementary contributors to the full model's lower control objective. Additional analysis results and ablation studies are provided in Appendix D.3.

5 Conclusion

We have introduced OMFM, an online meta-learning framework for control under unknown and varying dynamics. OMFM learns transferable control from sequential environments through shared trajectory-derived context that adapts control generation and predictive candidate evaluation. With one observed trajectory, context-conditioned generation and predictor-based candidate ranking improve control without additional environment interactions for candidate selection. Experiments across five systems demonstrate superior performance and generalization to unseen dynamics.

AI USE STATEMENT

We used generative AI tools to help find and identify literature, develop and carry out research ideas, produce the authors' partial manuscript, and refine the language when preparing this work. We did not use AI tools to create synthetic datasets, to prove mathematical statements. All AI-assisted content was carefully reviewed and verified by the authors, who take full responsibility for the final content, including all text, claims, and experimental results presented in this paper.

REPRODUCIBILITY STATEMENT

We provide resources to facilitate reproducibility of this work. The code repository linked in the abstract provides implementations of OMFM and baseline methods, including complete training and evaluation scripts for the Pendulum dataset. The main paper describes the overall framework, experimental settings, and key results, while the appendix provides detailed descriptions of datasets, model architectures, training procedures, and additional experimental configurations. Together with the released code, these resources enable independent verification and reproduction of our reported results.

REFERENCES

Deepak Akhare, Luning Sun, Xin-Yang Liu, Xiantao Fan, Timo Bremer, Ben Zhu, and Jian-Xun Wang. Offline reinforcement learning for fluid controls: Data-based multi-observational policy extraction. *arXiv preprint arXiv:2606.31025*, 2026.

Arthur Argenson and Gabriel Dulac-Arnold. Model-based offline planning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=OMNB1G5xzd4.

Lazar Atanackovic, Xi Zhang, Brandon Amos, Mathieu Blanchette, Leo J. Lee, Yoshua Bengio, Alexander Tong, and Kirill Neklyudov. Meta flow matching: Integrating vector fields on the wasserstein manifold. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=9SYczU3Qgm.

Andrew G. Barto, Richard S. Sutton, and Charles W. Anderson. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-13(5):834–846, 1983. doi: 10.1109/TSMC.1983.6313077.

Arjun Bhardwaj, Jonas Rothfuss, Bhavya Sukhija, Yarden As, Marco Hutter, Stelian Coros, and Andreas Krause. Data-efficient task generalization via probabilistic model-based meta reinforcement learning. *IEEE Robotics and Automation Letters*, 9(4):3918–3925, 2024.

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

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

S Scott Collis, Ronald D Joslin, Avi Seifert, and Vassilis Theofilis. Issues in active flow control: theory, control, simulation, and experiment. *Progress in aerospace sciences*, 40(4-5):237–289, 2004.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1329–1338, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/duan16.html.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online meta-learning. In *International conference on machine learning*, pp. 1920–1930. PMLR, 2019.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396. IEEE, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Lukas Hewing, Kim P Wabersich, Marcel Menner, and Melanie N Zeilinger. Learning-based model predictive control: Toward safe learning in control. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):269–296, 2020.

Philipp Holl, Vladlen Koltun, and Nils Thuerey. Learning to control pdes with differentiable physics. *arXiv preprint arXiv:2001.07457*, 2020a.

Philipp Holl, Vladlen Koltun, Kiwon Um, and Nils Thuerey. phiflow: A differentiable pde solving framework for deep learning via physical simulations. In *NeurIPS workshop*, volume 2. NeurIPS, 2020b.

Peiyan Hu, Xiaowei Qian, Wenhao Deng, Rui Wang, Haodong Feng, Ruiqi Feng, Tao Zhang, Long Wei, Yue Wang, Zhi-Ming Ma, et al. From uncertain to safe: Conformal adaptation of diffusion models for safe pde control. *arXiv preprint arXiv:2502.02205*, 2025.

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

Xueying Jiang, Min Huang, Huiyuan Shi, Xingwei Wang, and Yanfeng Zhang. Off-policy two-dimensional reinforcement learning for optimal tracking control of batch processes with network-induced dropout and disturbances. *ISA transactions*, 144:228–244, 2024.

Matthew Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM review*, 59(4):849–904, 2017.

Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21):e2101784118, 2021.

Athindran Ramesh Kumar and Peter J Ramadge. Diffloop: Tuning pid controllers by differentiating through the feedback loop. In *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6. IEEE, 2021.

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. Context-aware dynamics model for generalization in model-based reinforcement learning. In *International Conference on Machine Learning*, pp. 5757–5766. PMLR, 2020.

Jacob Levy, Jason Gibson, Bogdan Vlahov, Erica Tevere, Evangelos Theodorou, David Fridovich-Keil, and Patrick Spieler. Meta-learning online dynamics model adaptation in off-road autonomous driving. *arXiv preprint arXiv:2504.16923*, 2025.

Yanrong Li, Juan Du, and Wei Jiang. Reinforcement learning for process control with application in semiconductor manufacturing. *IISE Transactions*, 56(6):585–599, 2024a.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024b.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Yongjae Lim, Yeonsoo Park, Jeongtae Huh, Youngsoo Han, and H Jin Kim. Efficient learning-based path planning on receding horizon with conditional flow matching: Y. lim et al. *International Journal of Control, Automation, and Systems*, 24(4):666–677, 2026.

Yiheng Lin, James A Preiss, Fengze Xie, Emile Anand, Soon-Jo Chung, Yisong Yue, and Adam Wierman. Online policy optimization in unknown nonlinear systems. In *The Thirty Seventh Annual Conference on Learning Theory*, pp. 3475–3522. PMLR, 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Jianan Liu, Zike Zhou, Wenjing Hong, and Jia Shi. Two-dimensional iterative learning control with deep reinforcement learning compensation for the non-repetitive uncertain batch processes. *Journal of Process Control*, 131:103106, 2023.

Kai Liu, YangQuan Chen, Tao Zhang, Siyuan Tian, and Xi Zhang. A survey of run-to-run control for batch processes. *ISA transactions*, 83:107–125, 2018.

Danylo Malyuta, Yue Yu, Purnanand Elango, and Behçet Açıkmeşe. Advances in trajectory optimization for space vehicle control. *Annual Reviews in Control*, 52:282–315, 2021.

Carlos Mastalli, Wolfgang Merkt, Josep Marti-Saumell, Henrique Ferrolho, Joan Solà, Nicolas Mansard, and Sethu Vijayakumar. A feasibility-driven approach to control-limited ddp. *Autonomous Robots*, 46(8):985–1005, 2022.

Yu Mei, Xinyu Zhou, Shuyang Yu, Vaibhav Srivastava, and Xiaobo Tan. Fast online adaptive neural mpc via meta-learning. *IFAC-PapersOnLine*, 59(30):377–382, 2025.

Alejandro Murillo-González and Lantao Liu. Action flow matching for continual robot learning. *arXiv preprint arXiv:2504.18471*, 2025.

Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl. *arXiv preprint arXiv:1812.07671*, 2018a.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation*, pp. 7559–7566. IEEE, 2018b. doi: 10.1109/ICRA.2018.8463189.

Mohammadreza Nakhaei, Aidan Scannell, and Joni Pajarinen. Residual learning and context encoding for adaptive offline-to-online reinforcement learning. In *6th Annual Learning for Dynamics & Control Conference*, pp. 1107–1121. PMLR, 2024.

Seohong Park, Qiyang Li, and Sergey Levine. Flow q-learning. *arXiv preprint arXiv:2502.02538*, 2025.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5331–5340. PMLR, 2019. URL https://proceedings.mlr.press/v97/rakelly19a.html.

Allen Ren, Justin Lidard, Lars Ankile, Anthony Simeonov, Pulkit Agrawal, Anirudha Majumdar, Benjamin Burchfiel, Hongkai Dai, and Max Simchowitz. Diffusion policy policy optimization. In *International Conference on Learning Representations*, volume 2025, pp. 77288–77329, 2025.

Spencer M Richards, Navid Azizan, Jean-Jacques Slotine, and Marco Pavone. Control-oriented meta-learning. *The International Journal of Robotics Research*, 42(10):777–797, 2023.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in neural information processing systems*, 32, 2019.

Hersh Sanghvi, Spencer Folk, and Camillo Jose Taylor. Occam: Online continuous controller adaptation with meta-learned models. *arXiv preprint arXiv:2406.17620*, 2024.

Guanya Shi, Kamyar Azizzadenesheli, Michael O'Connell, Soon-Jo Chung, and Yisong Yue. Meta-adaptive nonlinear control: Theory and algorithms. *Advances in Neural Information Processing Systems*, 34:10013–10025, 2021.

Balasubrahmanyan Srinivasan and Dominique Bonvin. Controllability and stability of repetitive batch processes. *Journal of Process Control*, 17(3):285–295, 2007.

Hongfeng Tao, Junhao Zheng, Junyu Wei, Wojciech Paszke, Eric Rogers, and Vladimir Stojanovic. Repetitive process based indirect-type iterative learning control for batch processes with model uncertainty and input delay. *Journal of Process Control*, 132:103112, 2023.

Auwal Shehu Tijjani, Ahmed Chemori, and Vincent Creuze. A survey on tracking control of unmanned underwater vehicles: Experiments-based approach. *Annual Reviews in Control*, 54:125–147, 2022.

Long Wei, Peiyan Hu, Ruiqi Feng, Haodong Feng, Yixuan Du, Tao Zhang, Rui Wang, Yue Wang, Zhi-Ming Ma, and Tailin Wu. Diffphycon: A generative approach to control complex physical systems. *Advances in Neural Information Processing Systems*, 37:4090–4147, 2024.

Long Wei, Haodong Feng, Yuchen Yang, Ruiqi Feng, Peiyan Hu, Xiang Zheng, Tao Zhang, Dixia Fan, and Tailin Wu. Cl-diffphycon: closed-loop diffusion control of complex physical systems. In *International Conference on Learning Representations*, volume 2025, pp. 99509–99545, 2025.

E Weinan, Jiequn Han, and Jihao Long. Empowering optimal control with machine learning: A perspective from model predictive control. *IFAC-PapersOnLine*, 55(30):121–126, 2022.

Kehan Wen, Yutong Hu, Yao Mu, and Lei Ke. Mˆ 3pc: Test-time model predictive control using pretrained masked trajectory model. In *International Conference on Learning Representations*, volume 2025, pp. 37721–37743, 2025.

Songwei Wu, Zhiduo Jiang, Wandong Sun, Guanghu Xie, Yuteng Xie, Rui Zhao, Yang Liu, and Hong Liu. Cola-flow policy: Temporally coherent imitation learning via continuous latent action flow matching for robotic manipulation. *IEEE Robotics and Automation Letters*, 2026.

Lingjian Ye, Feifan Shen, and Hongwei Guan. Design of hybrid batch-to-batch and within-batch self-optimizing control structures for batch processes. *Journal of Process Control*, 113:1–17, 2022.

Haeun Yoo, Ha Eun Byun, Dongho Han, and Jay H Lee. Reinforcement learning for batch process control: Review and perspectives. *Annual Reviews in Control*, 52:108–119, 2021.

L Zabeo, Giuseppe Ambrosino, M Cavinato, Y Gribov, A Kavin, V Lukash, Massimiliano Mattei, Alfredo Pironti, JA Snipes, G Vayakis, et al. Overview of magnetic control in iter. *Fusion Engineering and Design*, 89(5):553–557, 2014.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Russ R. Salakhutdinov, and Alexander J. Smola. Deep sets. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html.

Kevin Zeng and Michael D Graham. Symmetry reduction for deep reinforcement learning active control of chaotic spatiotemporal dynamics. *Physical Review E*, 104(1):014210, 2021.

Shuang Zhang, Xinyu Qian, Zhijie Liu, Qing Li, and Guang Li. Pde modeling and tracking control for the flexible tail of an autonomous robotic fish. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(12):7618–7627, 2022.

Tonghe Zhang, Chao Yu, Sichang Su, and Yu Wang. Reinflow: Fine-tuning flow matching policy with online reinforcement learning. *Advances in Neural Information Processing Systems*, 38: 106282–106319, 2026.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Zifeng Zhuang, Kun Lei, Jinxin Liu, Donglin Wang, and Yilang Guo. Behavior proximal policy optimization. *arXiv preprint arXiv:2302.11312*, 2023.

A Related Work

A.1 Learning Across Batches for Physical System Control

Controlling complex physical systems frequently necessitates the use of time-varying input sequences to guide the system toward specific states or trajectories. These challenges are prevalent in diverse physical domains, including underwater vehicles, active flow control, and magnetically confined fusion plasmas (Tijjani et al., 2022; Collis et al., 2004; Zabeo et al., 2014). Such systems typically exhibit nonlinear dynamics, strongly coupled or high-dimensional state spaces, long-term dependencies in control actions, and physical constraints, all of which complicate effective trajectory-level control.

Many physical and industrial processes are characterized by repeated finite-duration runs rather than continuous operation. Batch processes naturally exhibit this structure, with transient operation repeated across successive runs and substantial variability arising along both the temporal and batch dimensions (Srinivasan & Bonvin, 2007; Yoo et al., 2021). This repeated structure has motivated extensive research on batch-to-batch and iterative learning control, where information collected from completed runs is reused to improve subsequent operation (Ye et al., 2022; Tao et al., 2023). Recent studies further consider non-repetitive uncertainties and learning-based control across the batch dimension, reflecting continued interest in exploiting experience accumulated over successive runs (Liu et al., 2023; Jiang et al., 2024). Semiconductor manufacturing provides a representative run-to-run setting, where process measurements from completed runs are used to adjust subsequent recipes or control decisions (Liu et al., 2018; Li et al., 2024a). More generally, each experiment, episode, manufacturing run, or physical shot yields a finite trajectory together with observations and control outcomes that can inform subsequent operation. We refer to each such completed run as a *batch*, distinguishing it from a minibatch used in stochastic optimization.

An additional challenge arises when successive batches are subject to varying dynamics resulting from changes in physical parameters, operating conditions, or other unobserved system properties. In such cases, enhancing control for a single repeated process is insufficient; instead, accumulated experience should facilitate transferable control across previously unseen dynamics. This context motivates the formulation of learning across batches as the progressive acquisition of reusable control knowledge from sequentially encountered environments.

A.2 Methods for Physical System Control

Classical control approaches formulate control directly using a dynamical model. Trajectory optimization computes control sequences under explicit dynamical constraints, often employing numerical methods such as direct collocation (Kelly, 2017). MPC repeatedly solves a finite-horizon optimization problem as new observations become available, providing feedback by replanning during execution. Although these methods are effective when an accurate model and sufficient computational resources are available, they typically require repeated numerical optimization and explicit access to system dynamics.

Learning-based methods mitigate reliance on explicit models by learning predictive models or amortizing components of the control computation. Probabilistic dynamics ensembles facilitate model-based reinforcement learning and predictive control while accounting for model uncertainty (Chua et al., 2018). Differentiable physics offers an alternative by propagating gradients through simulated system evolution, enabling direct optimization of planning and control networks for downstream physical objectives (Holl et al., 2020a). Reinforcement learning approaches instead learn policies or value functions from trajectories and have been applied to both low-dimensional dynamical systems and complex control problems. In these formulations, learned representations can replace or augment repeated physical optimization, although performance remains contingent on the training distribution and the quantity of available interaction data.

Generative models provide an alternative to deterministic control prediction by learning distributions over feasible trajectories or action sequences. Diffusion-based approaches have been applied to offline planning, observation-conditioned action generation, and high-dimensional physical control, including Diffuser, Diffusion Policy, DiffPhyCon, and subsequent extensions for closed-loop efficiency and safety-aware control (Janner et al., 2022; Chi et al., 2025; Wei et al., 2024; 2025; Hu et al., 2025). Flow Matching offers a related generative formulation through learned transport vector

fields (Lipman et al., 2023). Recent research has investigated flow-based policies for temporally coherent action generation and efficient trajectory optimization (Wu et al., 2026; Lim et al., 2026). Collectively, these methods demonstrate the effectiveness of generative modeling for representing and optimizing control distributions. OMFM extends this perspective by focusing on the progressive improvement of a shared generative controller using batches collected from heterogeneous dynamical environments.

A.3 Online Adaptation and Continual Control Learning

Adaptation to unfamiliar dynamics can be achieved by inferring latent system information, updating predictive models, or adjusting controller parameters using newly collected data. Context-aware dynamics models infer environment representations from recent transitions and use these representations to improve prediction and model-based control under changing dynamics (Lee et al., 2020). Meta-learning utilizes experience across multiple systems to develop transferable representations or priors that enable rapid adaptation with limited observations (Richards et al., 2023; Bhardwaj et al., 2024). These approaches differentiate between adapting behavior to currently observed dynamics and accumulating knowledge that supports adaptation to future environments. This distinction motivates a broad perspective on online learning in which effective control depends on both adapting to current dynamics and transferring knowledge across successive environments.

Online adaptation extends this concept by updating controllers, predictive models, or learned representations during deployment. In contrast, online meta-learning explicitly leverages experience accumulated across sequentially encountered environments (Sanghvi et al., 2024; Nakhaei et al., 2024; Shi et al., 2021; Lin et al., 2024). This method differs from conventional online control methods such as MPC, which replans in response to observations within a trajectory but does not necessarily update knowledge for future environments. Continual learning introduces a retention objective, employing mechanisms such as model reuse and replay to balance adaptation to new environments with the preservation of previously acquired behaviors (Nagabandi et al., 2018a; Rolnick et al., 2019).

Recent research has extended online improvement to generative policies, demonstrating that diffusion-based and flow-based controllers can be refined through reinforcement learning or continual interaction (Ren et al., 2025; Zhang et al., 2026; Park et al., 2025; Murillo-González & Liu, 2025). Related meta-flow formulations indicate that learned vector fields can transfer across different distributions (Atanackovic et al., 2025). OMFM integrates these strategies by continually updating a shared flow-based control generator across heterogeneous dynamics. Transition-based context enables immediate adaptation, while bounded replay preserves valuable historical experience. The objective is to enhance performance on the current system and accumulate transferable control knowledge to improve policy generation for previously unseen dynamics.

B Additional Theoretical and Methodological Details

This appendix expands the constructions in Secs. 2 and 3. Context trajectories have the same format in training and testing, but only training data contribute to parameter updates.

B.1 Context Representation and Conditional Regression

The empirical measure in Eq. 6 places one atom on each sampled trajectory, rather than on each individual transition. The encoder in Eqs. 8–9 acts on the corresponding finite set $\mathcal{C}_{e_i^{\mathrm{tr}}}$. Its temporal blocks process the ordered features $\mathbf{x}_{i,j,0}, \ldots, \mathbf{x}_{i,j,H_{i,j}-1}$ within trajectory j. Mean and max pooling over $j \in \mathcal{K}_i$ then make $c_i = \Phi_\psi(\mathcal{C}_{e_i^{\mathrm{tr}}})$ invariant to a permutation of the context trajectories.

For the linear interpolation in Sec. 2.3, the optimal population regression function under squared loss is

$$v^\star(w, \alpha \mid y) = \mathbb{E}[V_\alpha \mid W_\alpha = w, \alpha, y], \tag{18}$$

where w is a possible value of the interpolated variable W_α. This is the conditional mean of the endpoint velocity given the current interpolated state, flow time, and conditioning information. Section 3.2 applies this interpretation to control generation with $y = (c_i, g_{i,j})$.

The regression statement follows from the conditional least-squares projection identity. With the encoder fixed, write $\mathsf{Z} = (U_\alpha, \alpha, c_i, g_{i,j})$ and $v^\star(\mathsf{Z}) = \mathbb{E}[V_\alpha \mid \mathsf{Z}]$, assuming $\mathbb{E}\|V_\alpha\|_2^2 < \infty$. For any square-integrable vector field $v(\mathsf{Z})$,

$$\mathbb{E}\|v(\mathsf{Z}) - V_\alpha\|_2^2 = \mathbb{E}\|v(\mathsf{Z}) - v^\star(\mathsf{Z})\|_2^2 + \mathbb{E}\|V_\alpha - v^\star(\mathsf{Z})\|_2^2. \tag{19}$$

The cross term vanishes because $\mathbb{E}[V_\alpha - v^\star(\mathsf{Z}) \mid \mathsf{Z}] = 0$. Thus the population minimizer among unrestricted fields is $v^\star(\mathsf{Z})$; minimizing over a restricted network class seeks its best L^2 approximation. Recovering the field conditioned on the latent environment ξ_i additionally requires the context representation to preserve the information about the conditional coupling of (U_0, U_1) relevant to control. Equation 19 characterizes the population regression target of the conditional flow.

B.2 Finite Control Targets and Loss Construction

The Gibbs measure in Eq. 10 describes the desired preference for low-cost controls. The finite training pool in Eq. 3 contains only trajectories indexed by $\mathcal{A}_i$. It supplies the same $n_i - n_c$ trajectories to the control flow and the dynamics predictor; no context trajectory is used as a control target or prediction label. For the flow, a uniform empirical proposal over the non-context condition–control pairs is tilted by their observed costs:

$$\begin{aligned} \widehat{q}_i(dg, dU) &= \frac{1}{|\mathcal{A}_i|} \sum_{j \in \mathcal{A}_i} \delta_{(g_{i,j}, U_{i,j})}(dg, dU), \\ \widehat{\mu}_i^\beta(dg, dU) &= \sum_{j \in \mathcal{A}_i} \widetilde{w}_{i,j}^\beta \delta_{(g_{i,j}, U_{i,j})}(dg, dU), \end{aligned} \tag{20}$$

where the weights are defined in Eq. 12 and sum to one over $\mathcal{A}_i$. This is a Gibbs tilt of the empirical joint distribution of (g, U). When each condition has one observed control, Eq. 20 provides a cost-weighted empirical training surrogate for the population conditional target $\nu_{\xi_i}^\beta(\cdot \mid g)$.

For each $j \in \mathcal{A}_i$, independently draw $U_0 \sim \gamma(\cdot \mid g_{i,j})$ and $\alpha \sim \mathrm{Unif}[0,1]$, set $U_1 = U_{i,j}$, and use the interpolation in Sec. 3.2. An explicit Gibbs-weighted generator loss is

$$\mathcal{L}_i^{\mathrm{gen}}(\psi, \theta) = \sum_{j \in \mathcal{A}_i} \widetilde{w}_{i,j}^\beta \mathbb{E}_{U_0, \alpha} \left[\|v_\theta(U_\alpha, \alpha \mid c_i, g_{i,j}) - (U_{i,j} - U_0)\|_2^2 \right], \tag{21}$$

where $c_i = \Phi_\psi(\mathcal{C}_{e_i^{\mathrm{tr}}})$ is shared across all non-context trajectories in environment i. The dataset, observed costs, context indices, and Gibbs weights are fixed when differentiating this loss. All non-context trajectories contribute, with lower-cost controls receiving higher weight.

B.3 Dynamics Supervision and Online Update

For an observed trajectory j in environment e_i^{tr}, the predictor receives $\mathrm{sg}[c_i]$. Its one-step predictions use the observed state at each step, whereas its recursive predictions feed predicted states back into f_ϕ. Writing $\widehat{s}_{i,j,t+1}^{\mathrm{one}}$ for the former and $\widehat{s}_{i,j,t}^{\mathrm{rec}}$ for the latter, the trajectory loss in Eq. 13 has the form

$$\begin{aligned} \mathcal{L}_{\mathrm{pred}} = \frac{1}{H_{i,j}} \sum_{t=0}^{H_{i,j}-1} & \ell_{\mathrm{state}}\big(\widehat{s}_{i,j,t+1}^{\mathrm{one}}, s_{i,j,t+1}\big) \\ & + \frac{\lambda_{\mathrm{multi}}}{|\mathcal{H}_{i,j}|} \sum_{h \in \mathcal{H}_{i,j}} \frac{1}{h} \sum_{t=1}^{h} \ell_{\mathrm{state}}\big(\widehat{s}_{i,j,t}^{\mathrm{rec}}, s_{i,j,t}\big), \end{aligned} \tag{22}$$

where $\mathcal{H}_{i,j} \subseteq \{1, \ldots, H_{i,j}\}$ is a nonempty set of prediction horizons and λ_{multi} weights the multi-horizon term. This expression specifies the two types of supervision; their precise normalization and pointwise loss ℓ_{state} may depend on the system. For length-40 pendulum trajectories, the pointwise loss is state-normalized smooth L_1, $\mathcal{H}_{i,j} = \{5, 10, 20, 40\}$, and $\lambda_{\mathrm{multi}} = 0.5$. The unweighted mean of Eq. 22 over $j \in \mathcal{A}_i$ gives $\mathcal{L}_i^{\mathrm{dyn}}$. The predictor receives $\mathrm{sg}[c_i]$ as an environment condition, but the context trajectories do not appear among its prediction targets.

Algorithm 1 OMFM Training and Deployment

Input: $\mathcal{E}_{\mathrm{tr}}$, $\mathcal{E}_{\mathrm{te}}$, B, n_c, β, $M = 16$.

Output: Trained parameters $\Theta_{N_{\mathrm{tr}}}$ and selected controls $\{\widehat{U}^*_{i,j}\}$.

// Stage 1: Online Amortized Meta-Learning

Initialize $\Theta = (\psi, \theta, \phi)$ and $\mathcal{R}_0 = \varnothing$.
for $i = 1, \ldots, N_{\mathrm{tr}}$ **do**
  Observe $\mathcal{D}_i$ and sample past-environment indices $\mathcal{B}_i$ from $\mathcal{R}_{i-1}$.
  for each optimization step **do**
    Sample contexts and complementary training pools for all $i' \in \{i\} \cup \mathcal{B}_i$; compute one shared $c_{i'} = \Phi_\psi(\mathcal{C}_{e^{\mathrm{tr}}_{i'}})$ per environment.
    Compute Gibbs-weighted generation and unweighted dynamics losses using $c_{i'}$ and $\mathrm{sg}[c_{i'}]$, respectively; then take one step on the objective in Eq. 14.
  end for
  Set $\Theta_i = \Theta$; update $\mathcal{R}_{i-1}$ with $\mathcal{D}_i$ to obtain $\mathcal{R}_i$, retaining at most B environment datasets.
end for

// Stage 2: Inference in Unseen Environments

Freeze $\Theta_{N_{\mathrm{tr}}}$.
for each $e^{\mathrm{te}}_i \in \mathcal{E}_{\mathrm{te}}$ **do**
  Encode its single context trajectory as $c_i = \Phi_\psi(\mathcal{C}_{e^{\mathrm{te}}_i})$.
  For each $g_{i,j}$, draw M independent controls from $\pi_\theta(\cdot \mid c_i, g_{i,j})$; predict and select using Eqs. 16 and 17.
  Execute open-loop with the selected $\widehat{U}^*_{i,j}$.
end for
return $\Theta_{N_{\mathrm{tr}}}$ and $\{\widehat{U}^*_{i,j}\}$.

The single-level objective underlying Eq. 4 averages over the current environment and sampled past environments:

$$\ell_i(\Theta) = \frac{1}{1 + |\mathcal{B}_i|} \sum_{i' \in \{i\} \cup \mathcal{B}_i} \mathbb{E}_{\mathcal{K}_{i'}} \left[\mathcal{L}\Big(\Theta; \mathcal{D}^{\mathrm{train}}_{i'}, \mathcal{C}_{e^{\mathrm{tr}}_{i'}}\Big) \right]. \tag{23}$$

The expectation is over sampled context indices $\mathcal{K}_{i'}$. Both the context set and its complementary training pool depend on this sample, as defined in Eqs. 2–3. The loss $\mathcal{L}$ combines control generation and dynamics prediction, with the weighting and gradient paths specified in Sec. 3.2. The replay buffer $\mathcal{R}_{i-1}$ stores trajectory data from at most B previously observed environments. Thus B bounds the number of stored environment datasets, rather than the total number of trajectories.

At training round i, sample past-environment indices $\mathcal{B}_i$ such that $\{\mathcal{D}_{i'} : i' \in \mathcal{B}_i\} \subseteq \mathcal{R}_{i-1}$. For each $i' \in \{i\} \cup \mathcal{B}_i$, access $\mathcal{D}_{i'}$, choose context indices $\mathcal{K}_{i'}$, and recompute the corresponding $c_{i'}$, non-context pool, Gibbs weights, and losses using the current shared parameters. For sampled contexts, Eq. 14 is a stochastic realization of Eq. 23 and is optimized using the prescribed Adam steps for each arriving environment. Because $\mathrm{sg}[c_{i'}]$ blocks the dynamics gradient at the encoder, $\nabla_\psi \mathcal{L}^{\mathrm{dyn}}_{i'} = 0$; the generator loss trains (ψ, θ) and the predictor loss trains ϕ. Replay contains only data from environments observed before round i, so the update uses only current and retained past data. The shared parameters undergo direct single-level optimization; each training round retains these parameters and replay data, then recomputes environment context for the next update.

B.4 Training and Deployment Procedure

Algorithm 1 summarizes training and deployment. The replay buffer stores trajectory data from at most B past environments.

The test context is used only in the forward passes of TACE, the control flow, and the dynamics predictor. Predicting all M candidate trajectories incurs model computation; physical interaction is required only for the selected control.

C Detailed Dataset and Experimental Settings

We consider five dynamical systems: Pendulum, CartPole, the Kuramoto–Sivashinsky equation (KS1D), Kolmogorov flow (K2D), and smoke transport (Smoke2D). Each environment specifies a fixed realization of the underlying dynamics. Learning proceeds over a randomly ordered stream of environments, using observed trajectories from the current environment and previously encountered environments. Physical parameters are withheld from the learning algorithms. The physical solvers are used to construct the trajectory datasets and evaluate the selected controls, rather than to provide additional rollouts during model optimization or candidate selection. Each training environment contains 25 trajectories with diverse control quality. Each test environment provides a single random context trajectory and 10 independent initial states, for which the model must generate control sequences before evaluation through a single ground-truth rollout.

We follow the main-text notation s_t and u_t for observations and controls, with dimensions d_s and d_u. The continuous physical time in the differential equations is denoted by t_{phys}, distinct from the flow time α and the discrete control-interval index t. For a trajectory with H control intervals, we write the state sequence as $s_{0:H}$ and the control sequence as $u_{0:H-1}$. These sequences correspond to $X = (s_0, \ldots, s_H)$ and $U = (u_0, \ldots, u_{H-1})$ in the main text. Thus, H controls correspond to $H+1$ stored states, and u_t acts over the interval from s_t to s_{t+1}. Reported performance is the resulting true objective J, not the predictor's score. Lower J indicates better performance within each task; objective values are not directly comparable across tasks.

C.1 Pendulum

Simulation environment setup. We consider regulation of a damped, externally driven pendulum toward its downward equilibrium, as shown in Figure 5. Its dynamics are

$$\begin{aligned} \dot{\vartheta}(t_{\text{phys}}) &= \omega(t_{\text{phys}}), \\ \dot{\omega}(t_{\text{phys}}) &= -\omega_{\text{n}}^2 \sin\vartheta(t_{\text{phys}}) - d\omega(t_{\text{phys}}) + g_{\text{pend}} u(t_{\text{phys}}) + A_{\text{ext}} \sin t_{\text{phys}}, \end{aligned} \tag{24}$$

where ϑ is the angular displacement from the downward position, ω is angular velocity, ω_{n} is the natural-frequency parameter, d is damping, g_{pend} is the actuation gain, and A_{ext} is the amplitude of a disturbance with fixed angular frequency 1. The state is $s = (\vartheta, \omega)$, and the scalar control satisfies $|u_t| \le 2$. The environment-dependent parameters are defined over the following support:

$$(\omega_{\text{n}}, d, g_{\text{pend}}, A_{\text{ext}}) \in [0.75, 1.25] \times [0.05, 0.25] \times [0.75, 1.25] \times [0.05, 0.35]. \tag{25}$$

These intervals specify the parameter support used to construct diverse environments rather than independent uniform sampling distributions.

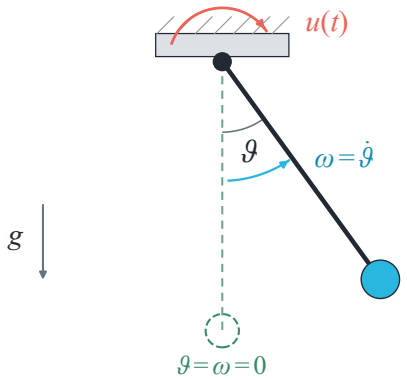


Figure 5: Pendulum control. An applied torque regulates the angular position and velocity toward the downward equilibrium.

Dataset. For each environment, the physical parameters are sampled from the above support. Initial conditions are sampled independently as

$$\vartheta_0 \sim \mathcal{U}(-1.5, 1.5), \qquad \dot{\vartheta}_0 \sim \mathcal{U}(-1, 1).$$

Experimental settings. Both state coordinates are directly observed, and no spatial observation mask is applied. The generated control trajectories are constrained within an action bound of 2. Since the pendulum dynamics integrator does not inherently enforce this constraint, the control-producing method is responsible for satisfying the admissibility requirement. For OOD evaluation, the environment dynamics are shifted by extending the natural frequency parameter beyond the training range. Specifically, OOD environments use $\omega_{\mathrm{n}} \in [0.65, 0.75) \cup (1.25, 1.35]$, while other physical parameters remain within their original ranges. This construction evaluates adaptation to unseen dynamical regimes while preserving the overall structure of the pendulum system.

Control objective. The objective penalizes displacement, angular velocity, control amplitude, and control variation over the complete trajectory:

$$J_{\mathrm{Pend}} = \frac{1}{H} \sum_{t=0}^{H-1} \left[\vartheta_{t+1}^2 + 0.1\omega_{t+1}^2 + 0.01u_t^2 + 0.002(u_t - u_{t-1})^2 \right], \qquad u_{-1} = 0. \tag{26}$$

The initial state is not included in the calculation of state penalties. The objective measures average regulation performance over the trajectory.

Implementation and evaluation. The simulator employs fixed-step fourth-order Runge–Kutta (RK4) integration with double precision. Each control input remains constant for 0.1 time units and is integrated using ten substeps of size 0.01. Each episode consists of 40 control intervals, spans 4 time units, and records 41 states. The held-out dataset supplies ten evaluation initial conditions for each environment. Table 4 summarizes the setup.

Table 4: Dataset and implementation settings for Pendulum.

Hyperparameter name	Value
State dimension / control dimension	2 / 1
Control horizon H / duration	40 / 4
Control interval / integration step	0.1 / 0.01
Numerical solver	RK4, double precision
Control bound	2
Training / ID test / OOD test environments	3000 / 50 / 50

C.2 CartPole

Simulation environment setup. The system consists of a cart of mass m_c and a uniform pole of mass m_p and half-length ℓ, as shown in Figure 6. The cart position is x, and the pole angle ϑ is measured from the upright equilibrium. Only the cart is actuated. With gravitational acceleration $g = 9.8$ and cart friction coefficient b, the equations of motion are

$$\begin{cases} r = \dfrac{F - b\dot{x} + m_p \ell \dot{\vartheta}^2 \sin\vartheta}{m_c + m_p}, \\ \ddot{\vartheta} = \dfrac{g \sin\vartheta - r\cos\vartheta}{\ell\,(4/3 - m_p \cos^2\vartheta/(m_c + m_p))}, \\ \ddot{x} = r - \dfrac{m_p \ell \ddot{\vartheta} \cos\vartheta}{m_c + m_p}. \end{cases} \tag{27}$$

The simulator integrates the physical state $(x, \dot{x}, \vartheta, \dot{\vartheta})$, while the stored observation and model input are

$$s = (x, \dot{x}, \cos\vartheta, \sin\vartheta, \dot{\vartheta}) \in \mathbb{R}^5. \tag{28}$$

During control interval t, the applied force is $F = \mathrm{clip}(u_t, -10, 10)$.

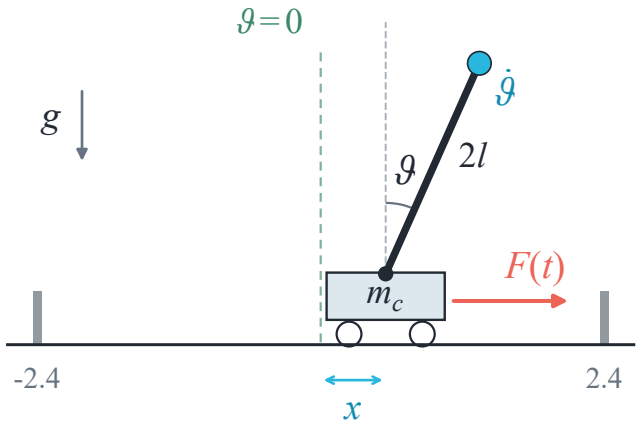


Figure 6: Cart–pole control. A horizontal force drives the cart toward the track center while stabilizing the pole upright.

Dataset. The initial physical state is sampled uniformly from

$$x_0 \in [-0.25, 0.25], \quad \dot{x}_0 \in [-0.5, 0.5], \quad \vartheta_0 \in [-0.15, 0.15], \quad \dot{\vartheta}_0 \in [-0.5, 0.5]. \tag{29}$$

This task involves stabilization from near-upright initial conditions. Physical-state initial conditions are preserved for simulation, while the corresponding five-dimensional observations are provided to the models.

Experimental settings. All five observation coordinates are fully observable, with no partial observation mask applied. The scalar force remains constant during each control interval, and the pole does not receive direct actuation. The nominal episode duration is set to one second. When the cart reaches an end stop, the rollout receives the terminal score defined below, and its score remains part of the reported evaluation results. For OOD evaluation, environment dynamics are altered by extending one physical parameter beyond the ID range, while the remaining parameters remain within their original ranges. Specifically, ID environments are generated by independently sampling the cart mass $m_c \in [0.5, 2.0]$, pole mass $m_p \in [0.05, 0.20]$, pole half-length $\ell \in [0.25, 0.75]$, and friction coefficient $b \in [0, 0.40]$. For each OOD environment, one parameter is randomly selected and sampled from its corresponding extended range: $m_c \in [0.4, 0.5) \cup (2.0, 2.2]$, $m_p \in [0.04, 0.05) \cup (0.20, 0.23]$, $\ell \in [0.20, 0.25) \cup (0.75, 0.85]$, or $b \in (0.40, 0.50]$. This approach introduces dynamic shifts by exposing the controller to previously unseen physical regimes while preserving the inverted pendulum system's overall structure.

Control objective. We set the target position to $x_\star = 0$ and the rail-limit scale to $x_{\lim} = 2.4$. For control u_t, the per-interval objective is

$$\begin{aligned} L_t = 1 - \cos\vartheta_{t+1} + \left(\frac{x_{t+1} - x_\star}{x_{\lim}}\right)^2 \\ + 0.05\left[\left(\frac{\dot{x}_{t+1}}{2}\right)^2 + \left(\frac{\dot{\vartheta}_{t+1}}{2}\right)^2\right] + 0.01\left(\frac{u_t}{10}\right)^2. \end{aligned} \tag{30}$$

If $t_{\text{stop}} \in \{0, \dots, H-1\}$ is the index of the first control interval whose resulting state contacts an end stop, the reported objective is

$$J_{\text{CP}} = \begin{cases} \sum_{t=0}^{H-1} L_t, & \text{if no end stop is reached,} \\ \sum_{t=0}^{t_{\text{stop}}-1} L_t + 1, & \text{otherwise.} \end{cases} \tag{31}$$

The sum is empty when $t_{\text{stop}} = 0$. A fixed cost of 1 is assigned to the contact interval, while subsequent intervals are excluded from consideration. The objective function is computed as a sum rather than a time average. The stored initial state does not contribute to the score. The control penalty is based on the supplied command; therefore, all methods must adhere to the force bound prior to evaluation, even though the simulator also enforces force clipping.

Implementation and evaluation. We use double-precision RK4 with a physics step of 0.005 and a control interval of 0.025, giving five integration substeps per control and 200 substeps per episode. There are 40 controls and 41 stored observations. Relevant settings are listed in Table 5.

Table 5: Dataset and implementation settings for CartPole.

Hyperparameter name	Value
Physical state / observation / control dimension	4 / 5 / 1
Control horizon H / duration	40 / 1
Control interval / integration step	0.025 / 0.005
Numerical solver	RK4, double precision
Gravitational acceleration / force bound	9.8 / 10
Training / ID test / OOD test environments	3000 / 50 / 50

C.3 1D Kuramoto–Sivashinsky Equation

Simulation environment setup. We consider the controlled Kuramoto–Sivashinsky equation on the periodic domain $x \in [0, 2\pi)$:

$$\frac{\partial w}{\partial t_{\text{phys}}} = -w\frac{\partial w}{\partial x} - \frac{\partial^2 w}{\partial x^2} - \kappa\frac{\partial^4 w}{\partial x^4} + \sum_{j=0}^{d_u-1} u_j(t_{\text{phys}})\chi_j(x). \tag{32}$$

Here $w(x, t_{\text{phys}})$ denotes the physical field, while u_j denotes control component j. The nonlinear and second-derivative coefficients remain constant, whereas κ varies between environments. The field is discretized using $N_x = 128$ uniformly spaced grid points. The model state comprises the complete 128-dimensional field, and $d_u = 32$ localized actuators are implemented. Each actuator profile is a wrapped, unit-peak Gaussian centered at $x_j = 2\pi j/32, j \in \{0, 1, \dots, 31\}$ with a width of $\sigma = \pi/32$. These profiles are not mean-subtracted or normalized to unit root-mean-square amplitude. The nominal coefficient bound is $|u_{t,j}| \le 2$.

Dataset. Training and in-distribution environments use $\kappa \sim \text{LogUniform}(0.15, 0.45)$, while out-of-distribution environments extend to the adjacent intervals $[0.10, 0.15)$ and $(0.45, 0.50]$. Initial fields are constructed as

$$\widetilde{w}_0(x) = \sum_{m=1}^{6} c_m \sin(mx + \vartheta_m), \qquad c_m \sim \mathcal{N}(0, m^{-4}). \tag{33}$$

Random phases are applied, followed by subtraction of the spatial mean and normalization to unit root-mean-square amplitude. Here m^{-4} denotes the variance, and the standard deviation provided to the generator is m^{-2}. Both full- and partial-observation experiments utilize the same set of underlying trajectories.

Experimental settings. The full-observation and partial-observation settings utilize identical underlying trajectories, as shown in Figure 7. In the full-observation setting, all 128 state values are provided to the models. In contrast, the partial-observation setting restricts observations to grid indices 0 to 31 and 96 to 127, resulting in 64 observed points. We denote the partial observation by $o_t = P_{\text{obs}} s_t$, where P_{obs} selects these grid indices; model inputs and prediction targets use o_t in place of s_t. In both settings, evaluation is conducted using the complete physical field. The forcing is specified by 32 prescribed actuator profiles, and the control configuration remains consistent across observation settings. During trajectory generation, controls are constrained by the sampler, with the effective run-specific bound established during dataset calibration. This approach enables the evaluation of adaptation under limited state observations while maintaining consistent underlying dynamical trajectories. To evaluate robustness, measurement and control noise are introduced separately by perturbing the observations and executed controls with a noise magnitude of 20%. These noisy conditions assess the robustness of adaptation to uncertainties in state estimation and control execution, while maintaining the underlying system dynamics.

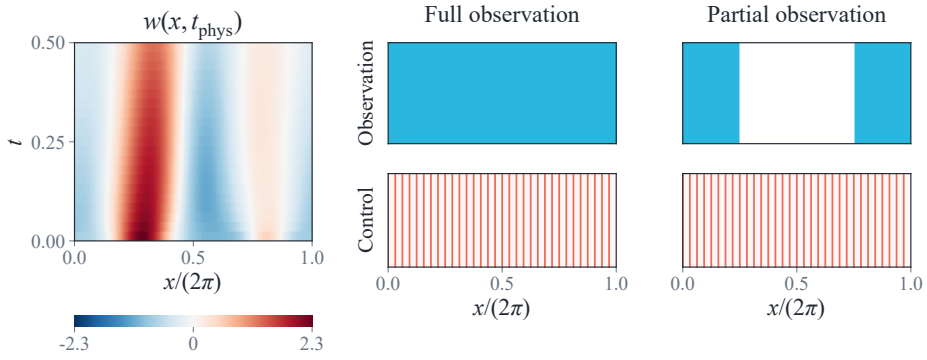


Figure 7: One-dimensional Kuramoto–Sivashinsky control. Distributed Gaussian actuators regulate the periodic state field under full or partial observation.

Control objectives. For the dissipation setting, the objective is the terminal mean-square field:

$$J_{\text{KS}} = \frac{1}{N_x} \sum_{i=0}^{N_x - 1} w_H(x_i)^2. \tag{34}$$

There is no control-energy penalty in the default configuration. For a partial observation mask $m_i \in \{0, 1\}$, the observation-based ranking functional is

$$\widehat{J}_{\text{KS,obs}} = \frac{\sum_i m_i \widehat{w}_H(x_i)^2}{\sum_i m_i}. \tag{35}$$

With full observation, the functional scores the entire predicted field; with partial observation, it averages squared predicted values over the observed region selected by the observation mask. For the dissipation setting, reported J uses Eq. 34 on the true rollout.

For spectrum matching, denote the orthonormally normalized discrete Fourier transform of w_H by $\widetilde{w}_H^{\text{ortho}}(k)$. Define

$$E_k = \frac{2}{N_x} \left| \widetilde{w}_H^{\text{ortho}}(k) \right|^2, \qquad E_k^\star = \frac{k^{-4}}{\sum_{r=1}^{6} r^{-4}}. \tag{36}$$

The configured spectrum objective is

$$J_{\text{KS,spec}} = \frac{1}{6} \sum_{k=1}^{6} (E_k - E_k^\star)^2. \tag{37}$$

Only modes 1–6 enter this objective; the zero and Nyquist modes are excluded. This setting is under full observation, and its reported score is $J_{\text{KS,spec}}$.

Implementation and evaluation. We use fourth-order exponential time-differencing Runge–Kutta (ETDRK4) integration with linear symbol $L_k = k^2 - \kappa k^4$ and two-thirds-rule dealiasing. Each of the 20 control intervals has a duration of 0.025 and contains 250 internal steps of size 10^{-4}. The total duration is 0.5, and each trajectory consists of 21 states. At this horizon, the task evaluates control of transient evolution from the prescribed initial fields. Table 6 presents the numerical settings.

Table 6: Dataset and implementation settings for KS1D.

Hyperparameter name	Value
Spatial domain / grid size	$[0, 2\pi)$ / 128
Boundary condition	Periodic
Physical state / observed state / control dimension	128 / 128 (full), 64 (partial) / 32
Control horizon H / duration	20 / 0.5
Control interval / integration step	0.025 / 10^{-4}
Numerical solver	ETDRK4, two-thirds dealiasing
Actuator width / nominal coefficient bound	$\pi/32$ / 2
Training / ID test / OOD test environments	3000 / 50 / 50

C.4 2D Kolmogorov Flow

Simulation environment setup. Kolmogorov flow is governed by the vorticity formulation of incompressible Navier–Stokes dynamics on the doubly periodic domain $(x, y) \in [0, 2\pi)^2$:

$$\frac{\partial \omega}{\partial t_{\text{phys}}} + \mathbf{v} \cdot \nabla \omega = \nu_{\text{vis}} \Delta \omega + F_0 \cos(k_f y) + \sum_{j=1}^{d_u} u_j(t_{\text{phys}}) \chi_j(x, y), \qquad \mathbf{v} = (\partial_y \Psi, -\partial_x \Psi), \qquad -\Delta \Psi = \omega - \langle \omega \rangle. \tag{38}$$

Here $\langle \cdot \rangle$ denotes the spatial average. The environment is defined by viscosity ν_{vis}, background-forcing amplitude F_0, and forcing wavenumber k_f. The field is discretized on a 64×64 grid. The state comprises 4096 vorticity values, while the control consists of 1024 coefficients arranged on a 32×32 lattice of actuator centers, as shown in Figure 8. Each profile is represented by a wrapped, unit-peak Gaussian with a width equal to half the center spacing, $\sigma = \pi/32$. Profiles are neither mean-subtracted nor root-mean-square normalized. Actuation is introduced additively in the vorticity equation and does not serve as a boundary condition. The nominal coefficient bound is 0.9.

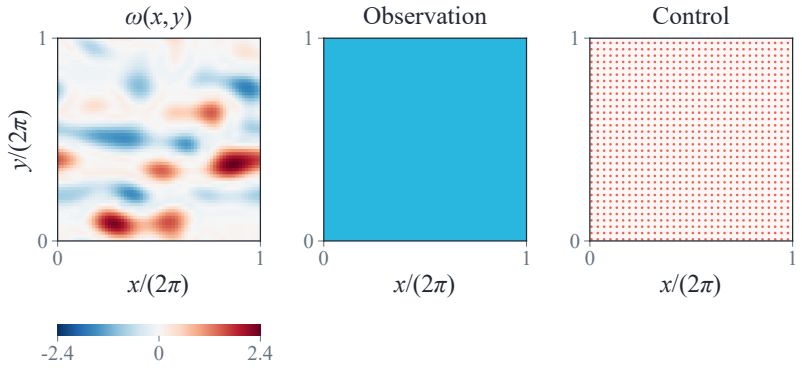


Figure 8: Two-dimensional Kolmogorov flow control. Distributed Gaussian actuators regulate the flow on a periodic domain with full-field vorticity observations.

Dataset. The training and in-distribution parameter distributions are

$$\nu_{\text{vis}} \sim \mathcal{U}(5 \times 10^{-4}, 8 \times 10^{-4}), \qquad F_0 \sim \mathcal{U}(0.75, 1.0), \qquad k_f \in [3, 4]. \tag{39}$$

Initial vorticity fields are random superpositions of Fourier modes with $1 \le |k_x| + |k_y| \le 4$. Their coefficients are standard Gaussian and their phases are random. Each field is made mean-free and

normalized to unit root-mean-square amplitude. The initial-state construction is independent of the environment parameters.

Experimental settings. The models receive the complete vorticity field as input, with actuation represented by a fixed 32×32 Gaussian basis. Evaluation is conducted over the entire 64×64 physical field, which contains 4096 grid points. For OOD evaluation, the dynamics distribution is altered by extending either the viscosity coefficient or the forcing amplitude beyond the ID range. Specifically, ID environments are generated by uniformly sampling ν_{vis} from $[5 \times 10^{-4}, 8 \times 10^{-4}]$ and forcing amplitude F_0 from $[0.75, 1.0]$, while each environment has a fixed forcing wavenumber $k_f \in [3, 4]$. OOD environments are divided into two equally sized groups: viscosity-shift environments sample ν_{vis} from $[4.5 \times 10^{-4}, 5 \times 10^{-4}) \cup (8 \times 10^{-4}, 8.5 \times 10^{-4}]$ with F_0 within the ID range, and forcing-shift environments sample F_0 from $[0.65, 0.75) \cup (1.0, 1.1]$ with ν_{vis} within the ID range. The admissible wavenumbers remain $k_f \in \{3, 4\}$ for ID and $k_f \in \{2.5, 4.5\}$ OOD environments; k_f is constant within each environment. This experimental design assesses adaptation to single-parameter shifts in dynamics while maintaining the Kolmogorov flow system.

Control objective. We minimize terminal enstrophy,

$$J_{\text{K2D}} = \frac{1}{2N_x N_y} \sum_{i=0}^{N_x-1} \sum_{j=0}^{N_y-1} \omega_H(x_i, y_j)^2, \qquad N_x = N_y = 64. \tag{40}$$

The default control-energy coefficient is set to zero, corresponding to a dissipative flow-control setting without explicit penalties on control magnitude. The initial normalization sets the initial enstrophy to $1/2$, and Eq. 40 evaluates terminal enstrophy.

Implementation and evaluation. The simulator employs double-precision pseudo-spectral spatial differentiation, two-thirds-rule dealiasing, and RK4 time integration with the viscous term evaluated on the right-hand side. Controls remain constant for 0.05 time units, with 500 internal steps of size 10^{-4} per interval. Each episode consists of $H = 20$ intervals, a total duration of 1, and 21 stored states. The spatial mean is subtracted at each recorded control interval. The periodic streamfunction relation is understood for the mean-free vorticity: for a field with nonzero mean, it reads $-\Delta\Psi = \omega - \langle\omega\rangle$, where $\langle\cdot\rangle$ denotes the spatial average. The controller enforces the coefficient bound, whereas the simulator does not impose any restriction on arbitrary input coefficients. Run-specific calibration may supersede the nominal bound. Table 7 provides a summary of the setup.

Table 7: Dataset and implementation settings for K2D.

Hyperparameter name	Value
Spatial domain / grid size	$[0, 2\pi)^2$ / 64×64
Boundary condition	Doubly periodic
State dimension / control dimension	4096 / 1024
Control horizon H / duration	20 / 1
Control interval / integration step	0.05 / 10^{-4}
Numerical solver	Pseudo-spectral RK4
Actuator lattice / width	32×32 / $\pi/32$
Nominal control coefficient bound	0.9
Training / ID test / OOD test environments	3000 / 50 / 50

C.5 2D Smoke Indirect Transport

Simulation environment setup. Smoke2D simulates indirect smoke transport toward a designated outlet in a flow domain with obstacles. Following CL-DiffPhyCon (Wei et al., 2025), we consider two settings: Domain control applies to both velocity components at the $1792 = 64^2 - 48^2$ locations outside the central 48×48 region of the 64×64 control-mask grid, whereas Boundary control is restricted to 32 prescribed boundary cells. They are shown in Figure 9. The corresponding control dimensions are 3584 and 64, both with amplitude bounds $u_{\max} = 2$. The simulator utilizes operator splitting: it constructs a prescribed velocity field, projects this field to enforce incompressibility, and

advects the smoke density. Let ϱ_t denote density, $\mathbf{v}_t$ velocity, u_t the control field supported on the admissible region, and $\mathbf{v}_{\text{amb}}$ a fixed ambient-circulation field. One control interval is represented by

$$\begin{cases} \widetilde{\mathbf{v}}_{t+1} = \mathcal{I}\mathbf{v}_t + \text{clip}(g_u\, u_t, -u_{\max}, u_{\max}) + \beta_{\text{circ}}\mathcal{I}\mathbf{v}_{\text{amb}}, \\ \mathbf{v}_{t+1} = \mathcal{P}(\widetilde{\mathbf{v}}_{t+1}), \\ \varrho_{t+1}^{\text{adv}} = \mathcal{A}(\varrho_t, \mathbf{v}_{t+1}), \end{cases} \tag{41}$$

where $\mathcal{I}$ denotes the simulator's fixed mapping of the stored and ambient velocity fields to the pre-projection velocity representation, $\mathcal{P}$ denotes the discrete pressure projection, $\mathcal{A}$ represents the advection operator, g_u is the hidden actuation gain, and β_{circ} is the hidden circulation coefficient. The captured outlet mass is measured on $\varrho_{t+1}^{\text{adv}}$ prior to removal. The remaining field is stored as ϱ_{t+1}. Training and ID environments sample $(g_u, \beta_{\text{circ}})$ independently and uniformly from $[0.9, 1.1]$ and $[-0.01, 0.01]$, respectively. OOD environments shift one parameter at a time beyond the corresponding range by 15% of its interval width, while keeping the other parameter within the original range. The actuation configuration remains fixed within each experiment. The density occupies a 63×63 cell-centered grid, while velocity is represented on a staggered Marker-and-Cell (MAC) grid. The model uses three channels on a common 64×64 storage grid, with the native density and velocity fields mapped to that storage representation. The stored observation dimension is therefore $d_s = 3 \times 64 \times 64 = 12288$; it is not the number of native density cells.

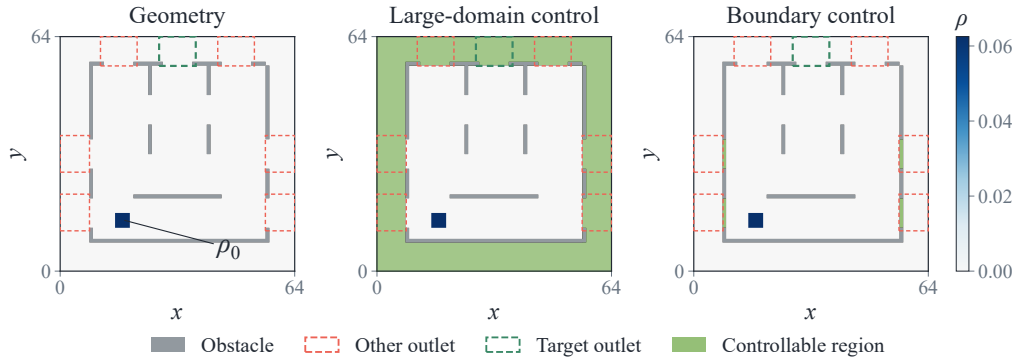


Figure 9: Two-dimensional smoke control. Velocity-field actuation guides smoke around obstacles toward the designated outlet, using either large-domain or boundary control.

Dataset. The initial smoke distribution is specified as a unit-mass 4×4 patch, with a density of $1/16$ assigned to each occupied cell. The patch anchor is sampled uniformly from wall-free locations with row indices in $\{11, 12, 13\}$ and column indices in $\{11, \ldots, 49\}$. The initial velocity is set to $(v_x, v_y) = (0, 1)$. The target region is defined as the top-middle outlet (exit 1), corresponding to rows 56 to 63 and columns 27 to 36 in the outlet mask. The executed state-action sequences are archived as trajectory data for model training and evaluation. Grid indices in this subsection are zero-based and inclusive. The outlet indices refer to the 64×64 mask representation, not directly to the 63×63 native density array; mass accounting uses the corresponding native-grid outlet region.

Experimental settings. We evaluate the two predefined actuation configurations, domain control and boundary control. The control configuration remains fixed throughout each experiment, allowing evaluation under different actuation constraints while preserving the same underlying smoke-transport dynamics.

Control objective. Let $S_{t,o}$ denote the smoke mass captured at outlet o during interval t, and let R_H be the remaining smoke mass after the final interval. We maximize the fraction routed to the target outlet $o_\star$ by minimizing

$$J_{\text{Smoke}} = 1 - \frac{\sum_{t=0}^{H-1} S_{t,o_\star}}{\sum_o \sum_{t=0}^{H-1} S_{t,o} + R_H}. \tag{42}$$

This is a trajectory-level objective with no control-energy term. The denominator is accumulated captured mass plus remaining mass, rather than the initial mass alone.

Implementation and evaluation. The simulator utilizes a Torch implementation of the reference PhiFlow (Holl et al., 2020b) execution, incorporating both the actuation-gain and ambient-circulation extensions. Each episode consists of 64 control intervals, each with a duration of 1, resulting in 65 stored frames. A single advection update occurs per control interval, without the use of a finer fixed integration step. The Domain and Boundary settings employ a shared state representation and objective, but differ in their respective actuation masks and control dimensions. Table 8 summarizes these distinctions.

Table 8: Dataset and implementation settings for Smoke2D. Paired entries refer to Domain and Boundary control, respectively.

Hyperparameter name	Value
Density grid / stored spatial grid	63×63 / 64×64
State channels / state dimension	3 / 12288
Control horizon H / stored frames	64 / 65
Control interval / episode duration	1 / 64
Controlled cells, Domain / Boundary	1792 / 32
Control dimension, Domain / Boundary	3584 / 64
Control bound, Domain / Boundary	2 / 2
Training / ID test / OOD test environments	2000 / 50 / 50

D Detailed Model and Experimental Settings

D.1 Models

We detail the implementation of each method. Let H represent the control horizon, d_s the observation dimension, and d_u the control dimension. These parameters define the input and output sizes. Unless otherwise specified, multilayer perceptrons (MLPs) consist of two hidden layers with a width of 96 andsigmoid linear unit (SiLU) activations. Control outputs are restricted to the admissible action range.

Our model Our implementation comprises the TACE Φ_ψ, a control flow-matching network v_θ, and a separately trained dynamics predictor f_ϕ. TACE processes transition features $[s_t, u_t, \Delta s_t]$, where $\Delta s_t = s_{t+1} - s_t$. Local feature embeddings are processed through temporal residual blocks, followed by temporal mean and max pooling. A subsequent pooling operation concatenates the element-wise mean and maximum of the trajectory representations across the context set, yielding a summary invariant to trajectory order. This is followed by an MLP and a non-affine LayerNorm to produce $c_i \in \mathbb{R}^{d_c}$, with $d_c = 16$. The vector encoder consists of two 96-unit feature layers. The spatial encoder employs a pointwise lifting layer, two Fourier blocks of width 32 that retain 8 modes per spatial axis, and spatial average pooling to construct per-transition features, followed by the same temporal processing and context aggregation.

The control generator takes a noisy control sequence of shape $H \times d_u$. A 1×1 temporal convolution projects it to 96 channels, and learned temporal embeddings encode the sequence positions. Four residual blocks use kernel-size-3 convolutions with dilation rates $(1, 2, 4, 8)$. Each block incorporates feature-wise linear modulation conditioned on the initial observation, flow time, and inferred context, together with single-group GroupNorm and SiLU activations. Two additional temporal residual blocks with dilation rates $(1, 2)$ precede the linear control head. The network is trained by conditional flow matching along $U_\alpha = (1 - \alpha)U_0 + \alpha U_1$, with target velocity $V_\alpha = U_1 - U_0$, Gaussian reference controls U_0, and target controls U_1 drawn using the finite-sample construction in the main-text methodology. Context trajectories are excluded from the control-target set. Inference uses 8 forward-Euler steps and generates 16 candidates per initial condition.

The dynamics predictor receives the current observation, control, physical-time information, and detached context $\mathrm{sg}[c_i]$. We write $f_{\phi,t}$ for its time-conditioned prediction at control interval t. The prediction loss updates ϕ only; the encoder and control generator are updated by the control-generation

loss. The vector predictor uses two 128-unit SiLU layers to estimate acceleration and advances the state through a kinematic update. For spatial observations, the reference predictor is a residual Fourier neural operator with 4 blocks, width 32, and 12 retained modes per spatial axis. Each block combines spectral and pointwise convolutions, Gaussian error linear unit (GELU) activation, and context-dependent feature modulation. The predictor is trained with one-step and multistep prediction losses. At inference, its predicted trajectories are evaluated using the analytic control objective, and the candidate with the lowest predicted cost is selected.

FMPhyCon FMPhyCon is the Flow-Matching variant of DiffPhyCon (Wei et al., 2024). It generates joint state–control trajectories conditioned on the initial observation, using the same 96-channel temporal architecture as our generator, with input and output widths changed to $d_s + d_u$. It predicts a conditional flow velocity and uses 8 Euler steps. Objective-gradient guidance is applied to an estimated clean trajectory, with reference guidance coefficient 0.1. The model conditions on the initial observation without an inferred environment context.

OFM OFM (Lipman et al., 2023) learns a global control-sequence distribution conditioned on the initial observation. Its velocity network uses 96 channels and four residual temporal blocks with dilation rates $(1, 2, 4, 8)$, eight-group GroupNorm, and SiLU activations. The conditioning MLP receives the initial observation and scalar, sine, and cosine flow-time features. Sampling uses 8 Euler steps. The implementation returns the first of the sampled candidates, with no learned dynamics-based selection.

Sequence BC and FTML-Policy The behavior-cloning (Zhao et al., 2023) implementation is a conditional variational autoencoder. An action-sequence encoder predicts a 32-dimensional Gaussian latent variable. Both the encoder and policy decoder contain 2 Transformer layers with embedding width 128, 4 attention heads, and feed-forward width 256. The decoder conditions on the initial observation and latent variable and predicts the entire control sequence. Training combines reconstruction and Kullback–Leibler (KL) losses with KL weight 1; deterministic inference sets the latent variable to zero. FTML-Policy (Finn et al., 2019) uses a two-hidden-layer, 96-unit MLP with a scaled $\tanh$ output. It performs one differentiable inner update with learning rate 10^{-2} and optimizes the resulting policy on held-out trajectories within each training environment. Its default evaluation uses the learned initialization directly, without a test-time gradient update.

SAC-offline, SAC-surrogate, and BPPO The reinforcement-learning policies receive the observation, normalized time, and previous control. For open-loop evaluation, the policy is unrolled using surrogate-predicted observations to obtain the complete control sequence before the physical rollout. Their actors use squashed diagonal Gaussian distributions, and their actor and value networks use the two-layer MLP configuration above. SAC (Haarnoja et al., 2018) uses twin Q-networks, automatic entropy-temperature adjustment, an undiscounted objective ($\gamma_{\mathrm{RL}} = 1$), and target-network updates with coefficient 0.005. SAC-offline learns from recorded transitions. SAC-surrogate additionally generates trajectories using the current learned dynamics model at each update. BPPO (Zhuang et al., 2023) fits a behavior policy, a Monte Carlo value function, and a state–action–reward–state–action (SARSA) Q-function before clipped policy improvement. Its clipping threshold starts at 0.25 and is multiplied by 0.96 at successive policy steps; the positive-advantage weight is 0.9.

M^{3}PC M^{3}PC (Wen et al., 2025) employs a bidirectional Transformer architecture applied to interleaved state and action tokens. The network comprises four layers with an embedding width of 192, 4 attention heads, a feed-forward width of 384, and no dropout. Learned mask, position, time-step, and modality embeddings are used to identify missing entries. Gaussian output heads are utilized to reconstruct masked states and controls, with a training mask ratio of 0.5. During planning, the model samples 16 candidates using path inference and inverse dynamics, performs three refinement rounds, and selects the candidate exhibiting the lowest forward-dynamics consistency error.

PID and Open-loop MPC The PID control law is bounded, and its gains are optimized through a learned dynamics surrogate (Kumar & Ramadge, 2021). We apply PID only to Pendulum and CartPole, as extending this implementation to spatially distributed systems would require an additional task-specific mapping from field errors to actuator inputs. MPC uses the cross-entropy method (CEM). It performs one CEM iteration at each predicted time step and warm-starts the next

plan by shifting the previous sequence. Both methods construct the final sequence using surrogate states. PID and MPC use a two-hidden-layer, 96-unit residual MLP for vector-valued dynamics, while MPC (Chua et al., 2018) uses the Fourier dynamics architecture for spatial observations. The default surrogate-based baselines do not infer an environment context.

D.2 Experimental Details

Computing environment. All experiments are implemented in Python with PyTorch on Linux GPU servers equipped with NVIDIA RTX 5090 GPUs (32 GB memory) and PyTorch 2.8.0 with CUDA 12.9. All models are trained on a single GPU. We generate datasets and perform training with three independent random seeds, and report the mean performance across these runs. For reproducibility, all random generators are explicitly seeded and deterministic training settings are enabled.

Continual training. The reference training schedule processes the randomly ordered environment stream sequentially. The past-only buffer $\mathcal{R}_{i-1}$ stores at most $B = 256$ previously observed training environments. Each update uses the current environment and up to 3 past environments randomly sampled from this buffer; their previously observed trajectories are recovered for the update. All methods use the same task-specific number of optimization steps per environment: 1 step for Pendulum and CartPole, 3 steps for 1D KS, and 5 steps for K2D and Smoke2D. For OMFM, the generation and dynamics losses are combined with $\lambda_{\mathrm{dyn}} = 1$ and averaged across the sampled environments before each single-level Adam update, without a task-specific inner loop. We set $\beta = 10$ and use four context trajectories during training and one during evaluation. The TACE encoder and control generator use learning rate 10^{-3} for the first 100 environments and 5×10^{-5} thereafter. The dynamics predictor uses 10^{-3} for the first 500 environments and 3×10^{-4} thereafter. The baseline-suite default learning rate is 5×10^{-4}, with gradient-norm clipping at 10. SAC and BPPO use transition minibatches of size 256 and follow the same task-specific update schedule as the other methods; BPPO additionally performs its behavior/value fitting within each update.

Evaluation protocol. For OMFM, all learned parameters $\Theta = (\psi, \theta, \phi)$ remain fixed during evaluation. OMFM computes $c_i = \Phi_\psi(\mathcal{C}_{e_i^{\mathrm{te}}})$ from the single observed context trajectory and generates $M = 16$ candidates for each evaluation initial condition. The predictor simulates each candidate using the same context, and the candidate with the lowest predicted objective is selected. Each method constructs a complete, bounded control sequence prior to its scored physical rollout. Candidate generation, surrogate prediction, and selection utilize only information available to the controller. The selected sequence is executed once in the physical simulator, and performance is assessed by the analytic objective evaluated on the resulting trajectory. Physical rollouts are not used to select among candidate controls. All compared methods are evaluated from identical initial conditions under consistent observation and actuation constraints.

D.3 Additional Experimental Results

This section provides additional analyses of the update budget, objective-gradient guidance, trajectory availability, computational cost, and generalization to previously untested environments. We report the control objective J, for which lower values indicate better performance, and use the method names and the evaluation protocol consistent with those presented in the main text.

Number of Updates per Environment We vary the number of optimization steps per arriving environment for Pendulum from 1 to 5, with 1 step as the default. Table 9 demonstrates that additional updates consistently reduce J. These results indicate that OMFM benefits from further optimization of the available data, reducing J from 0.1366 to 0.1291. One update already achieves $J = 0.1366$, while five updates further reduce the objective using the same available trajectories. This experiment modifies training updates rather than test-time adaptation steps; the learned parameters remain fixed during evaluation.

Effect of Objective-Gradient Guidance and Context Encoder DiffPhyCon (Wei et al., 2024) combines a learned joint state–control distribution with control-objective guidance during sampling.

Table 9: Effect of optimization steps per environment on Pendulum. One step is the default.

Steps per environment	1 (default)	2	3	4	5
OMFM (Ours)	0.1366	0.1331	0.1307	<u>0.1301</u>	**0.1291**

Following the FMPhyCon implementation described in the main text, we investigate objective-gradient guidance applied to the estimated clean trajectory. Table 10 shows that the effect of guidance varies across the evaluated systems.

Table 10: Objective-gradient guidance ablation for FMPhyCon. Positive values favor guidance.

Method	Pendulum	CartPole	KS1D Dissipation	K2D	Smoke2D Domain
FMPhyCon w/ guidance	0.2321	15.4553	0.2822	0.3986	0.3239
FMPhyCon w/o guidance	0.2284	14.9764	0.3093	0.4065	0.3187
Relative improvement	-1.62%	-3.20%	8.76%	1.94%	-1.63%

To further investigate the impact of environment conditioning, we augment FMPhyCon with the context encoder introduced in OMFM, while maintaining its joint state-control generation framework. Table 11 shows that incorporating environment representations consistently enhances control performance across the evaluated systems. These findings suggest that trajectory-derived context offers valuable insights into underlying dynamics, thereby enabling generative control models to more effectively adapt to variations across environments.

These experiments support environment conditioning and examine objective guidance across the evaluated systems. First, the context encoder enables environment-aware conditioning for control generation, which allows the model to adapt its proposals based on observed dynamics. Second, the separation of generation from evaluation keeps objective gradients from predicted trajectories out of the sampling process. As a result, the dynamics predictor is utilized solely to rank generated candidates, while control generation remains independent of objective gradients derived from predicted trajectories.

Table 11: Effect of adding OMFM's context encoder to FMPhyCon. Relative improvement is measured against FMPhyCon without the encoder.

Method	Pendulum	KS1D Dissipation	K2D
FMPhyCon w/o encoder	0.2321	0.2822	0.3986
FMPhyCon w/ encoder	0.2093	0.2644	0.3728
Relative improvement	**9.82%**	**6.31%**	**6.47%**

Number of Trajectories per Environment We vary the number of available training trajectories per environment on Pendulum from 5 to 50. Table 12 shows that all compared methods benefit from additional trajectories, while OMFM consistently achieves the lowest J across all data budgets. The advantage of OMFM over FTML-Policy, the strongest baseline in this comparison, persists under both limited and sufficient trajectory availability. With 10 trajectories per environment, OMFM achieves $J = 0.1473$, below FTML-Policy's 0.1503 with 50 trajectories. These results indicate that OMFM extracts transferable control knowledge even when only a limited number of trajectories are available for each environment.

Computational Cost Analysis Table 13 presents a comparison of training and inference costs for KS1D Dissipation and Smoke2D Boundary. OMFM combines low control cost with faster candidate selection than Open-loop MPC. During deployment, model parameters remain fixed, and gradient computation is unnecessary. In the inference stage, OMFM generates multiple candidate controls

Table 12: Control performance on Pendulum with different numbers of training trajectories per environment.

Method	Trajectories per environment			
	5	10	25	50
PID	0.1971	0.1885	0.1870	0.1840
Open-loop MPC	0.2163	0.1959	0.1875	0.1778
SAC-surrogate	0.2062	0.1954	0.1783	0.1717
FTML-Policy	0.1724	0.1630	0.1547	0.1503
OMFM (Ours)	**0.1617**	**0.1473**	**0.1366**	**0.1307**
Improvement	**6.21%**	**9.63%**	**11.70%**	**13.04%**

and ranks them using the learned predictor, thereby avoiding additional physical rollouts. Compared with Open-loop MPC, its reported inference time decreases from 0.929 to 0.040 seconds on KS1D and from 91.630 to 9.420 seconds on Smoke2D Boundary.

Table 13: Training and inference costs on KS1D Dissipation and Smoke2D Boundary. Time is reported in seconds and GPU memory in GB. OMFM uses $M = 16$ candidates at inference.

Method	KS1D Dissipation				Smoke2D Boundary			
	Time (s)		Memory (GB)		Time (s)		Memory (GB)	
	Train	Infer	Train	Infer	Train	Infer	Train	Infer
Open-loop MPC	2553.3	0.929	1.263	0.070	13906.8	91.630	1.217	0.083
M^3PC	5711.4	0.169	1.031	0.070	16986.9	3.663	1.377	0.098
Sequence BC	2215.0	**0.001**	0.825	0.067	12639.9	**1.028**	1.115	**0.035**
SAC-offline	3126.7	0.058	1.173	0.068	10579.3	10.486	1.922	0.075
SAC-surrogate	3760.8	0.061	1.264	0.068	13492.7	5.869	2.058	0.075
BPPO	3325.6	0.064	**0.164**	0.068	**10549.2**	12.545	1.924	0.044
FTML-Policy	**2186.4**	0.050	0.795	0.065	12575.5	4.151	0.991	0.102
OFM	3235.6	0.011	0.873	0.066	20205.1	4.194	**0.988**	0.040
FMPhyCon	5807.2	0.053	0.895	0.073	34538.0	18.310	1.851	0.085
OMFM (Ours)	4136.6	0.040	2.015	**0.051**	22862.3	9.420	4.121	0.050

In-Distribution and Out-of-Distribution Results Table 14 presents evaluation results in ID and OOD environments across Pendulum, CartPole, KS1D Dissipation, K2D, and Smoke2D Domain control. The OOD split is defined by the parameter shifts specified in Appendix C. OMFM demonstrates superior performance in both ID and OOD settings, retaining its advantage when assessed on dynamics outside the training distribution. These findings suggest that the learned environment representation supports effective adaptation through both interpolation within the training range and generalization to previously unobserved dynamical variations.

D.4 Visualization Results

Qualitative Comparison Figures 10–21 visualize the control evolution of OMFM and baseline methods across different dynamical systems. The results show that OMFM generates control trajectories that better drive the systems toward the desired objectives, illustrating lower-cost state evolution in these selected comparisons with the baseline controllers. These qualitative observations are consistent with the quantitative improvements reported in the main experiments, illustrating how the measured objective improvements appear in state trajectories across the evaluated physical systems.

Additional OOD Cases Figures 22 and 23 present two OOD cases where OMFM exhibits degraded control performance under challenging distribution shifts. These examples indicate that, although OMFM generalizes well across moderate shifts, its performance can degrade under more challenging distribution changes.

Table 14: ID and OOD control performance across five dynamical systems. KS1D uses the Dissipation objective and Smoke2D uses Domain control.

Method	Pendulum		CartPole		KS1D		K2D		Smoke2D	
	ID	OOD	ID	OOD	ID	OOD	ID	OOD	ID	OOD
PID	0.1816	0.1924	10.1443	14.4060	–	–	–	–	–	–
Open-loop MPC	0.1837	0.1913	10.2443	14.9984	1.0668	1.2727	0.6045	0.6209	–	–
M^{3}PC	0.2774	0.3062	19.8083	20.4605	1.5209	1.7645	0.6447	0.6519	0.5065	0.5555
Sequence BC	0.1919	0.2051	18.8766	20.2443	0.4372	0.5798	0.2518	0.2535	0.2124	0.2053
SAC-offline	0.2199	0.2441	25.6358	27.2405	0.3974	0.5555	0.5168	0.5194	–	–
SAC-surrogate	0.1629	0.1937	11.1373	15.4925	0.1289	0.2325	0.7387	0.7341	0.1986	0.2105
BPPO	0.3191	0.3255	19.6174	21.3185	0.8355	0.9897	0.5188	0.5290	0.4810	0.5192
FTML-Policy	0.1532	0.1562	11.6206	17.9142	0.2272	0.3396	0.2275	0.2343	0.1578	0.1625
OFM	0.3133	0.3487	17.2332	20.6053	0.7622	0.9751	0.4478	0.4490	0.2993	0.2785
FMPhyCon	0.2196	0.2446	13.9149	16.9957	0.2147	0.3498	0.3942	0.4030	0.3176	0.3301
OMFM (Ours)	**0.1303**	**0.1428**	**9.3859**	**11.2617**	**0.0856**	**0.1708**	**0.1867**	**0.1895**	**0.0928**	**0.1144**

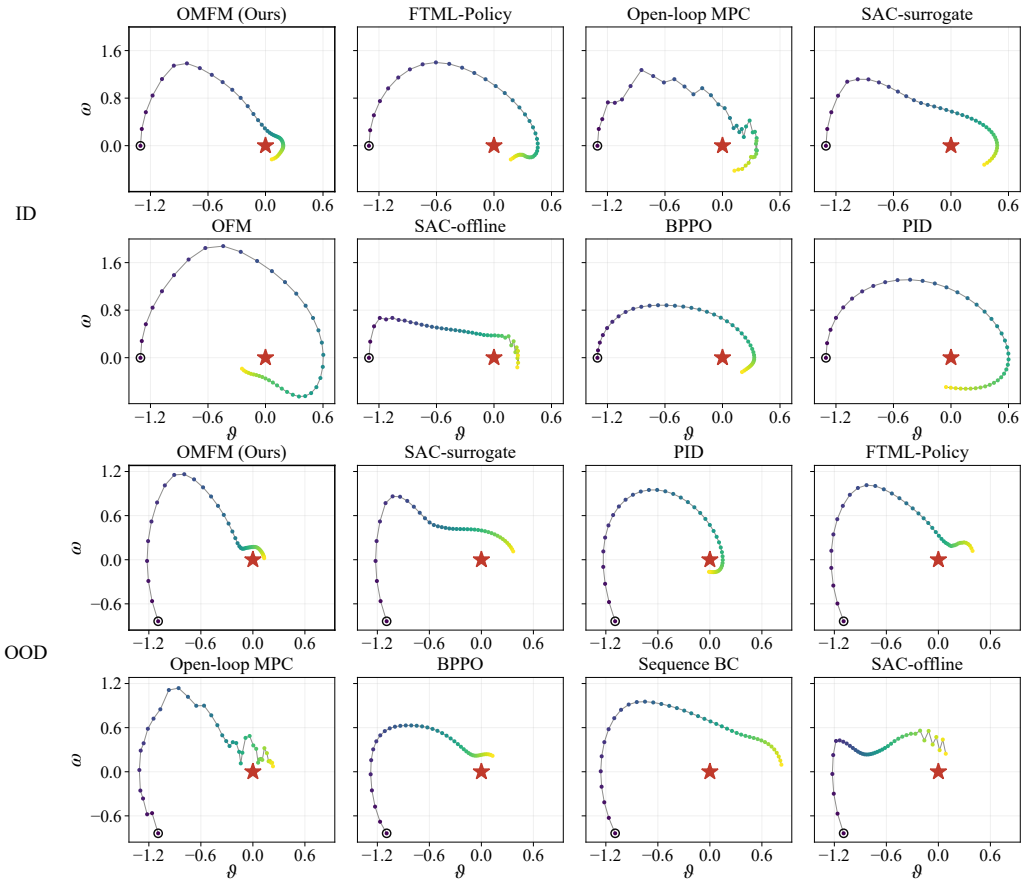


Figure 10: Comparison of OMFM and baseline controllers on ID and OOD pendulum regulation cases. Phase portraits show angular position and velocity, with time-colored trajectories, initial-state circles, and target-equilibrium stars.

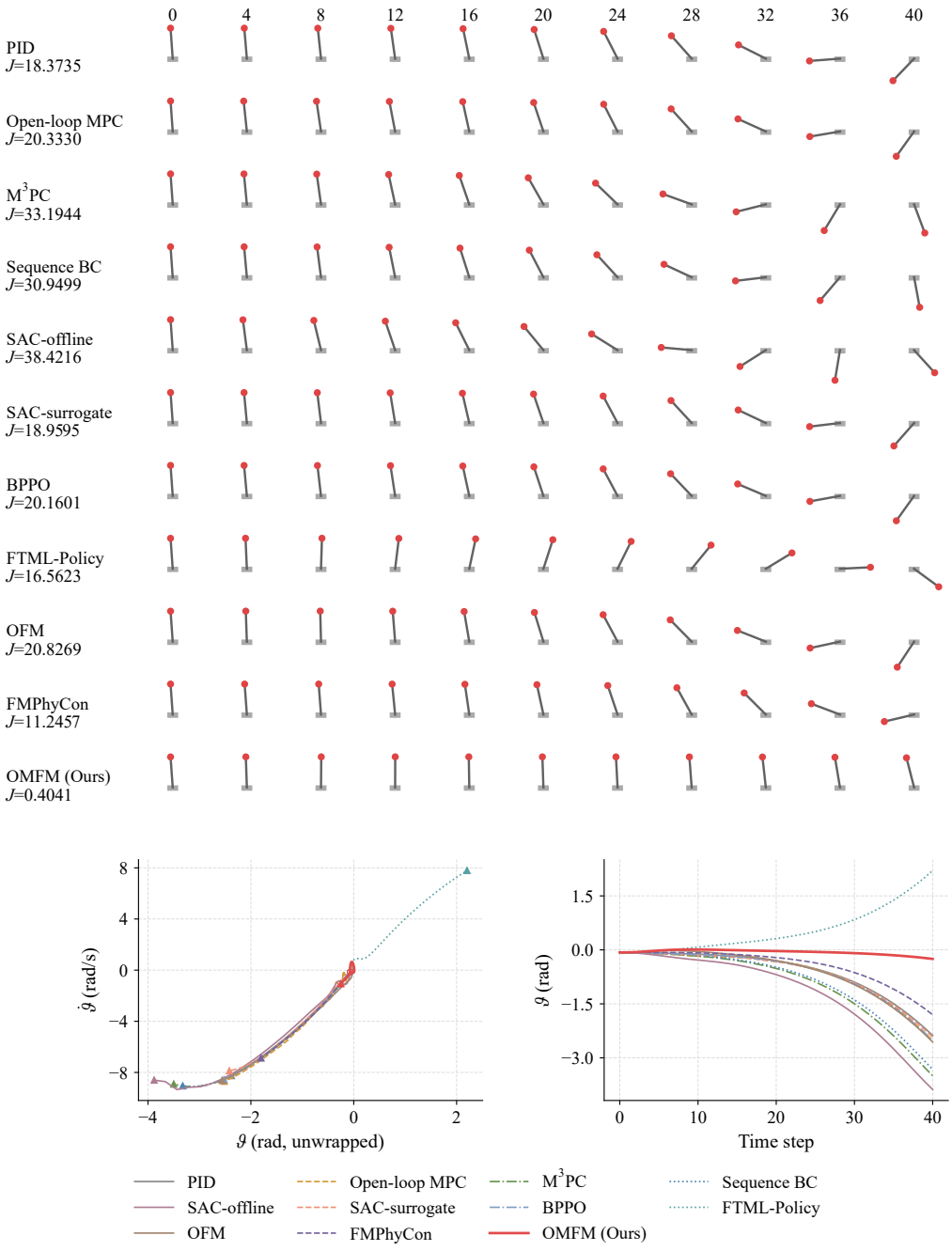


Figure 11: Comparison of OMFM and baseline controllers on the ID cartpole dataset, a benchmark for inverted-pendulum stabilization. Panels show pendulum snapshots, phase trajectories, and pole-angle evolution over the control horizon.

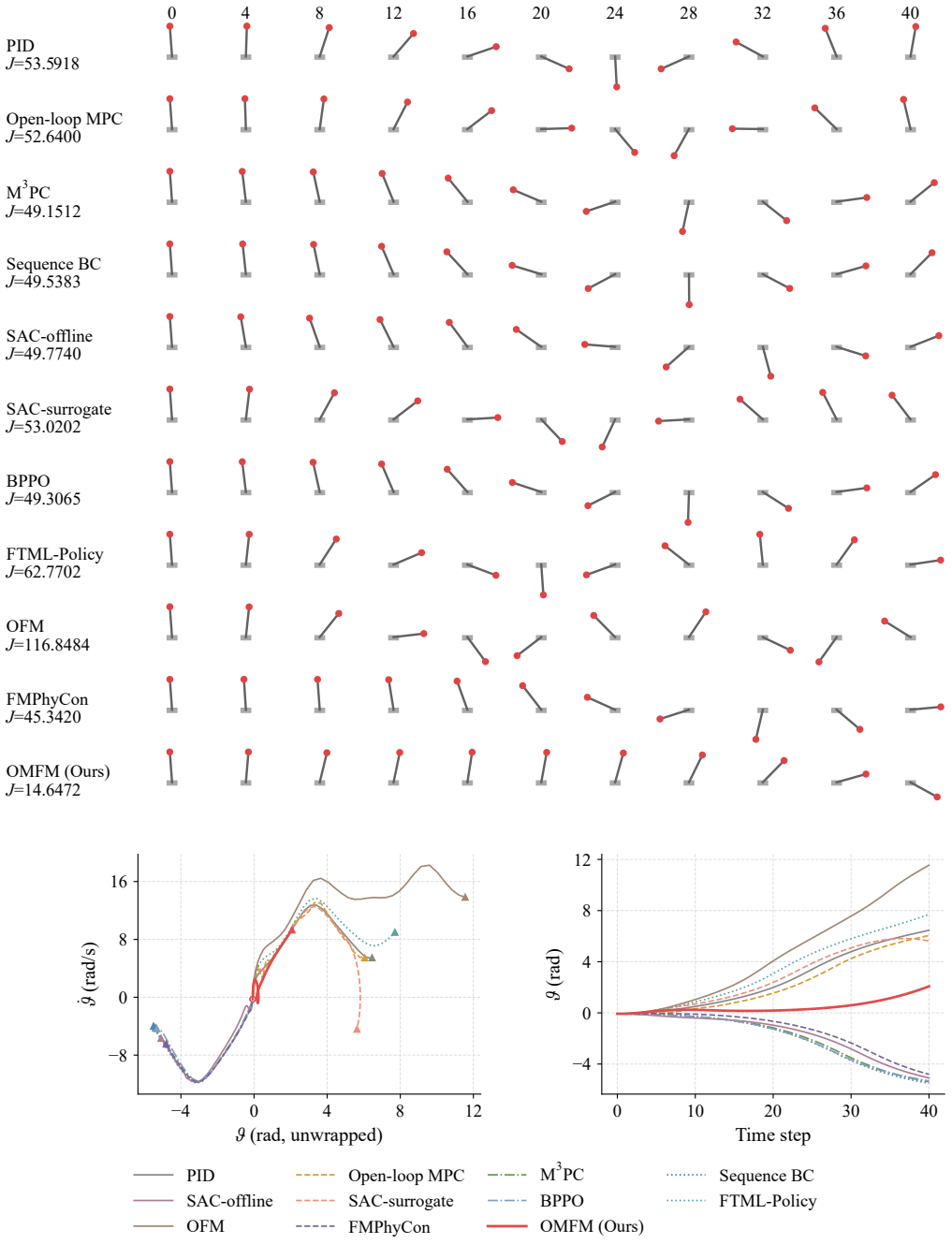


Figure 12: Comparison of OMFM and baseline controllers on the OOD cartpole dataset, a benchmark for inverted-pendulum stabilization. Panels show pendulum snapshots, phase trajectories, and pole-angle evolution over the control horizon.

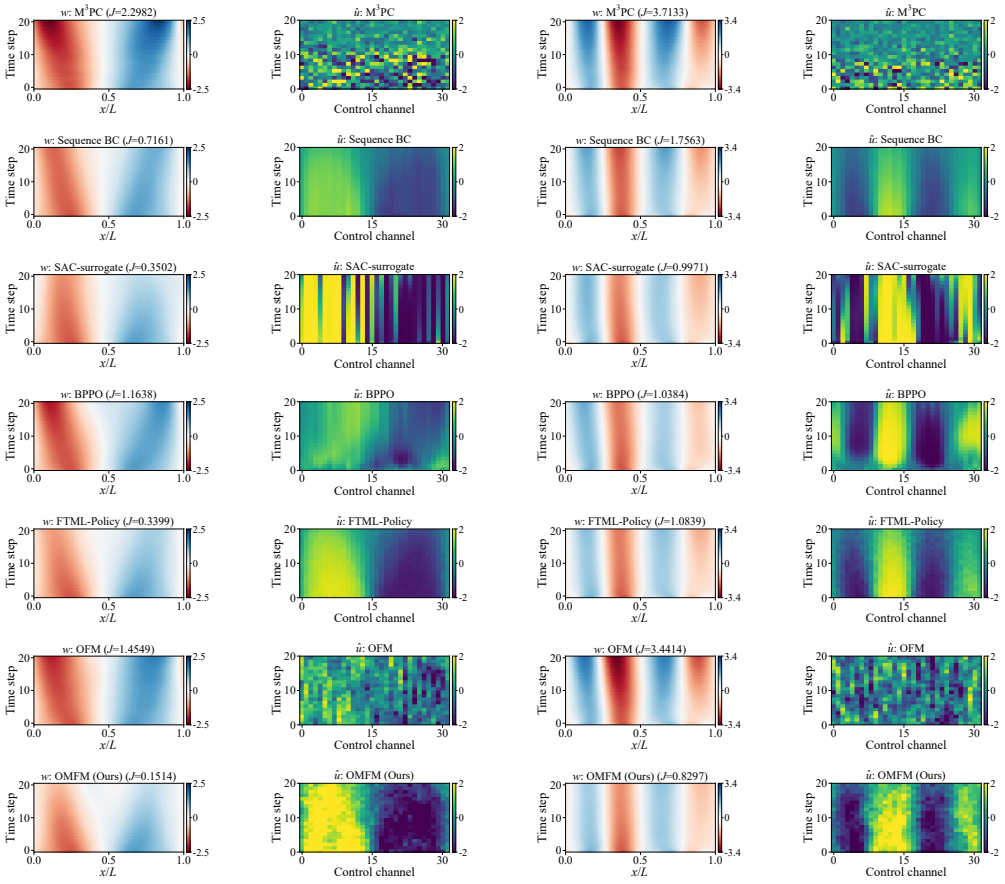


Figure 13: Comparison of OMFM and baseline methods on the OOD KS1D dataset, which describes nonlinear spatiotemporal dynamics. Paired columns show state evolution and applied controls for two selected test cases.

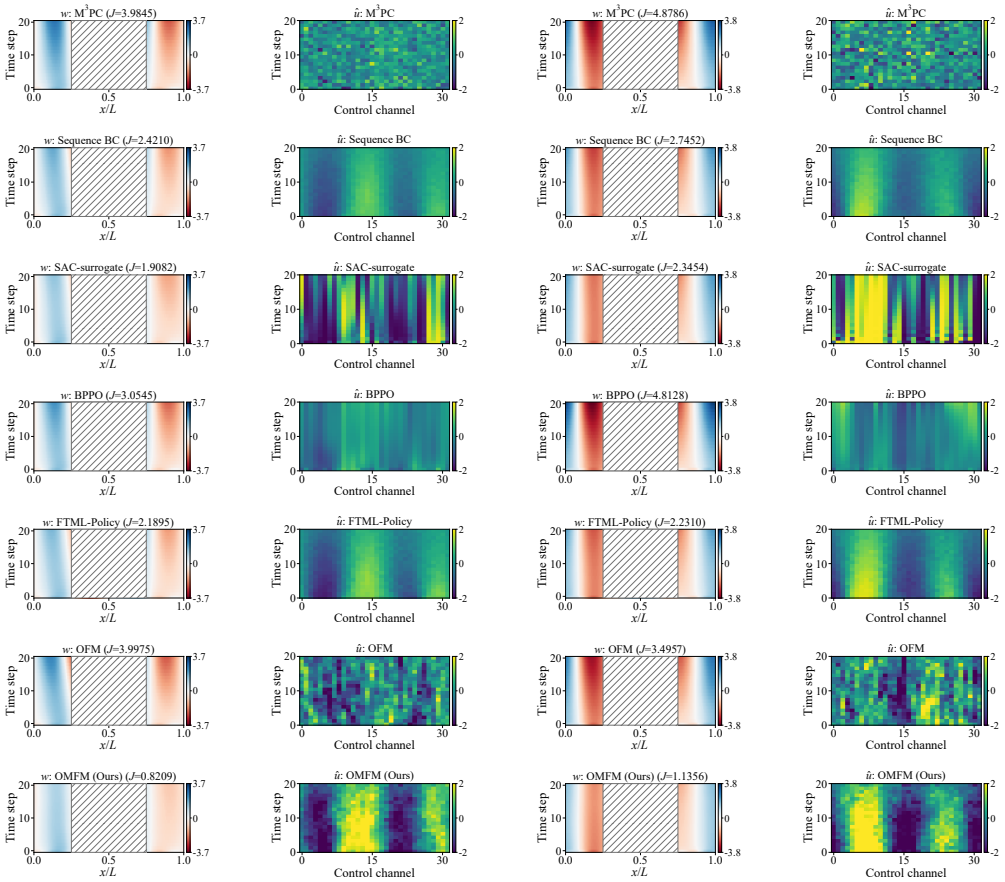


Figure 14: Comparison of OMFM and baseline methods on the OOD KS1D dataset under partial observation. Panels show state evolution and controls for two cases; hatching indicates the unobserved central spatial region.

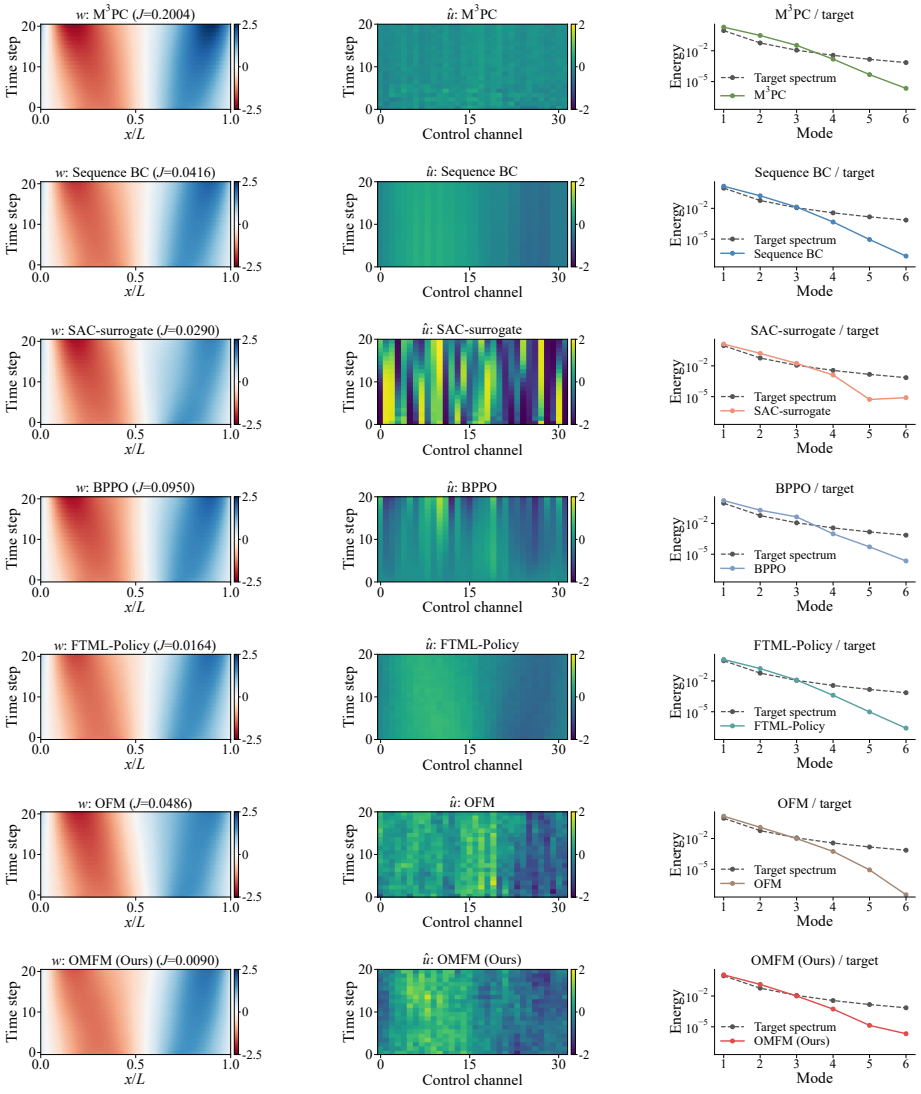


Figure 15: Comparison of OMFM and baseline methods on the OOD KS1D spectrum-control dataset, which targets prescribed modal energies. Each case includes state evolution, applied controls, and comparisons with the target spectrum.

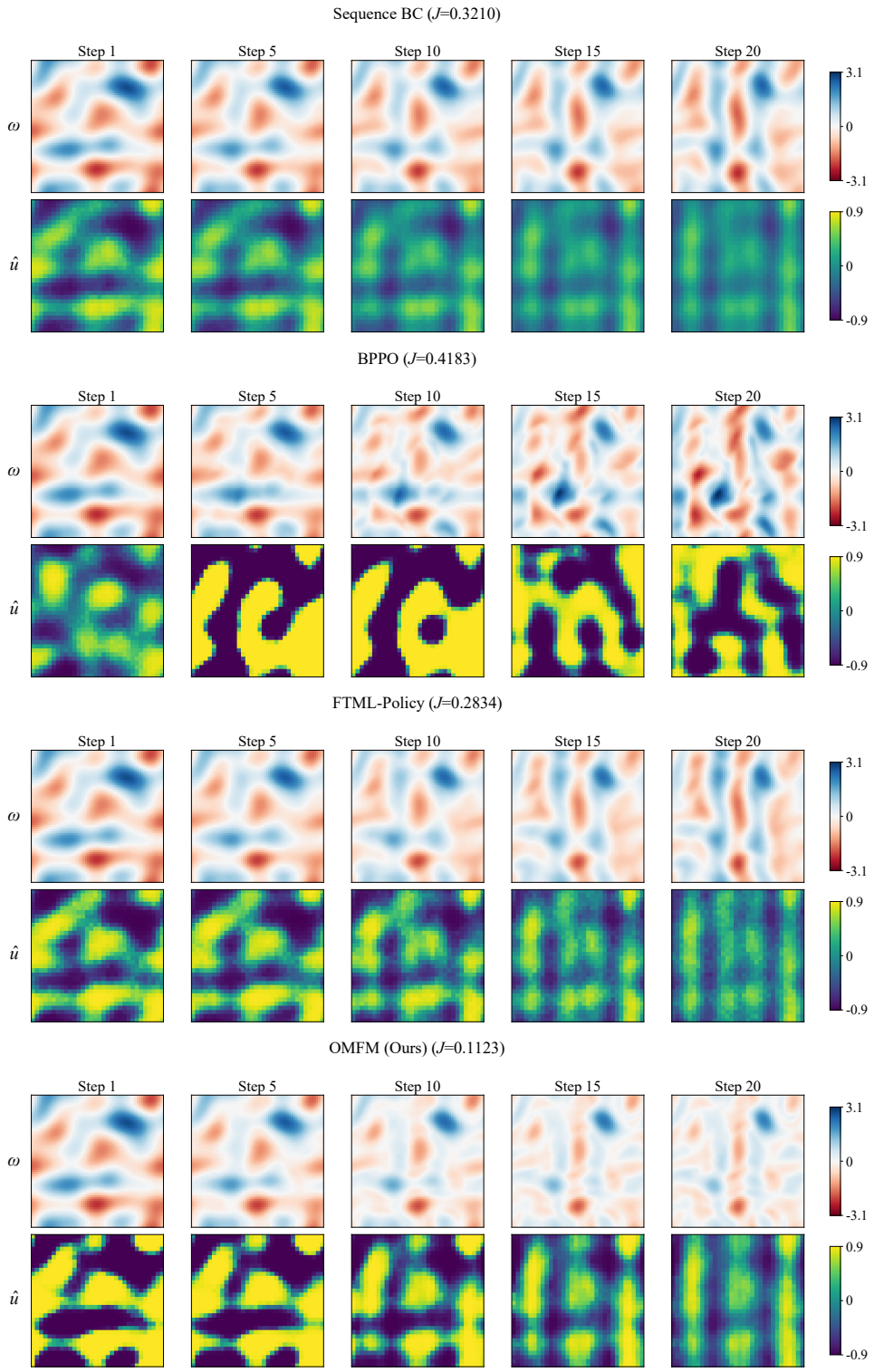


Figure 16: Comparison of OMFM and baseline methods on the ID K2D dataset, a two-dimensional fluid-control benchmark. Paired rows show state and control fields at five selected steps across the control horizon.

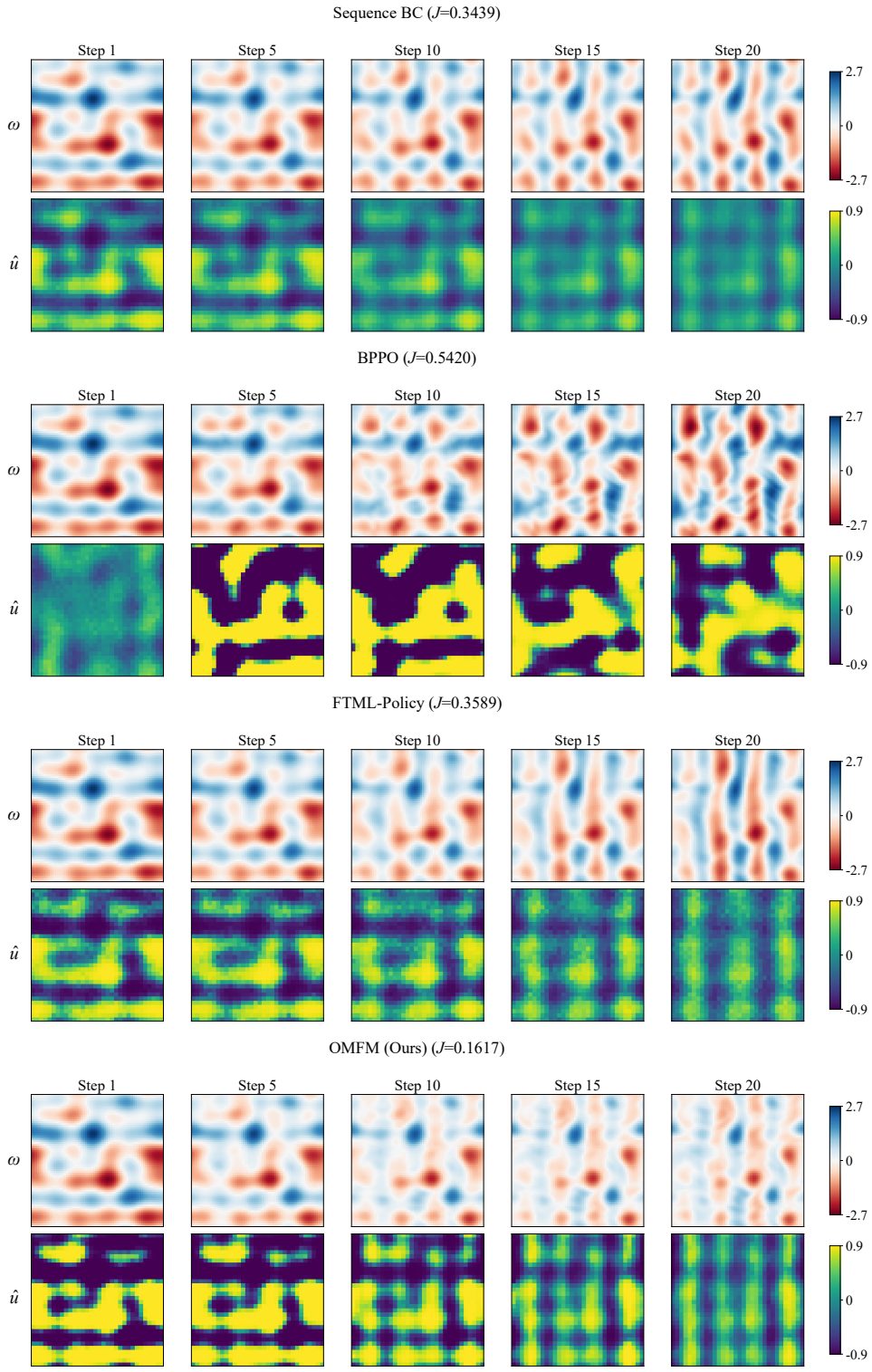


Figure 17: Comparison of OMFM and baseline methods on the OOD K2D dataset, a two-dimensional fluid-control benchmark. Paired rows show state and control fields at five selected steps across the control horizon.

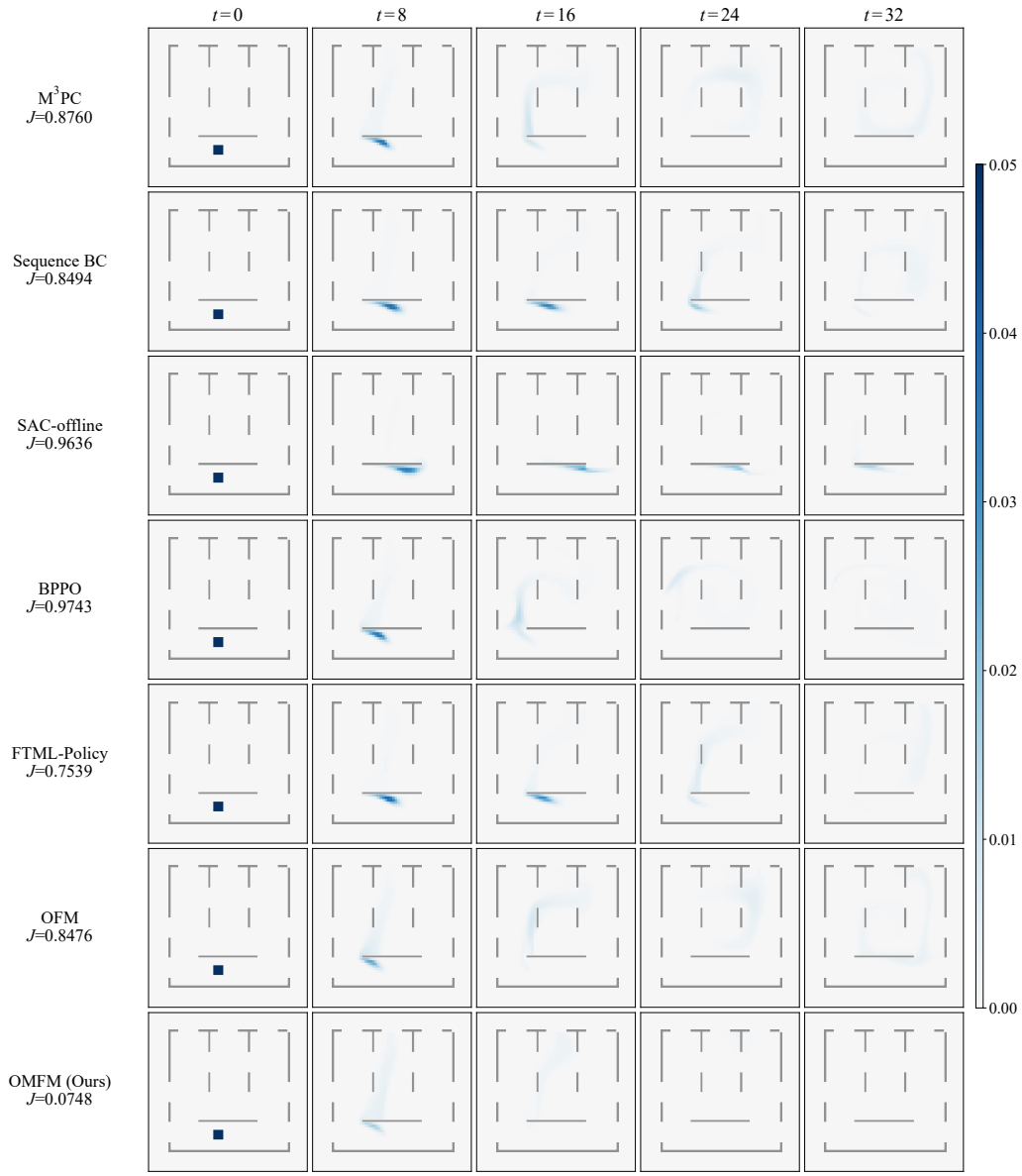


Figure 18: Comparison of OMFM and baseline methods on the ID Smoke2D dataset with boundary control, where smoke is guided toward a target outlet. Rows show density evolution through a fixed obstacle layout.



Figure 19: Comparison of OMFM and baseline methods on the OOD Smoke2D dataset with boundary control, where smoke is guided toward a target outlet. Rows show density evolution through a fixed obstacle layout.

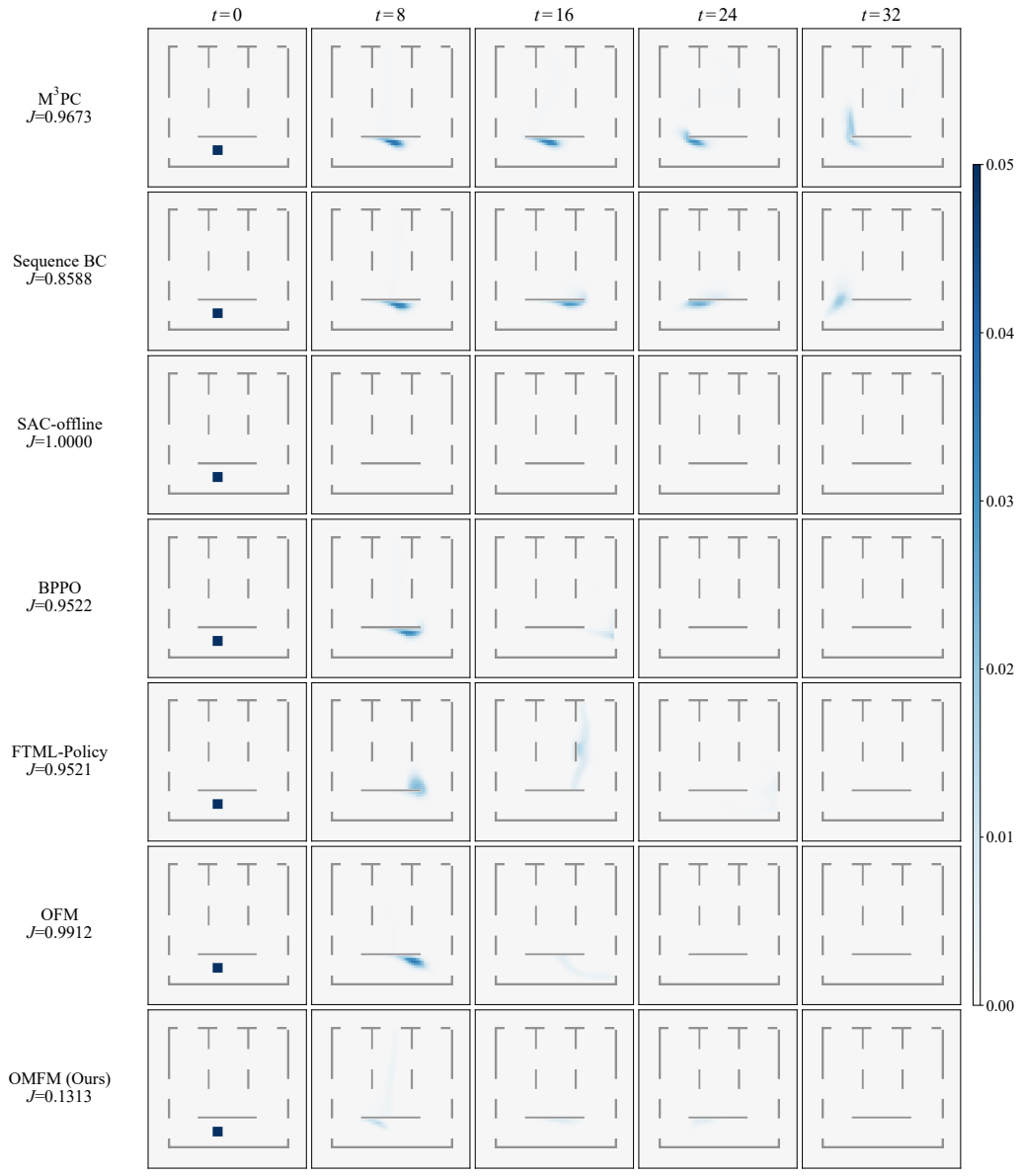


Figure 20: Comparison of OMFM and baseline methods on the ID Smoke2D dataset with a larger control region. The task guides smoke toward a target outlet; rows show density evolution around fixed obstacles.

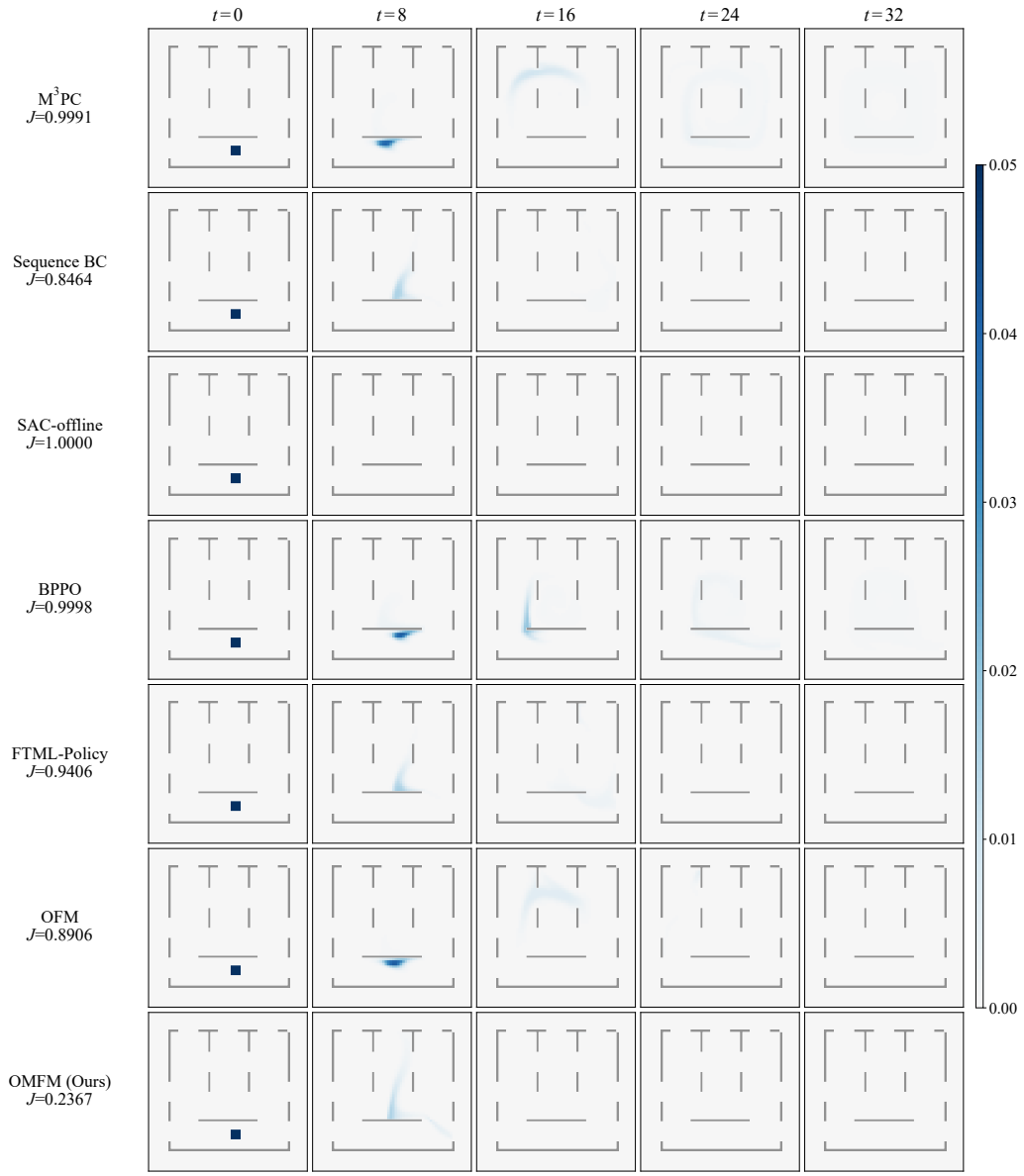


Figure 21: Comparison of OMFM and baseline methods on the OOD Smoke2D dataset with a larger control region. The task guides smoke toward a target outlet; rows show density evolution around fixed obstacles.

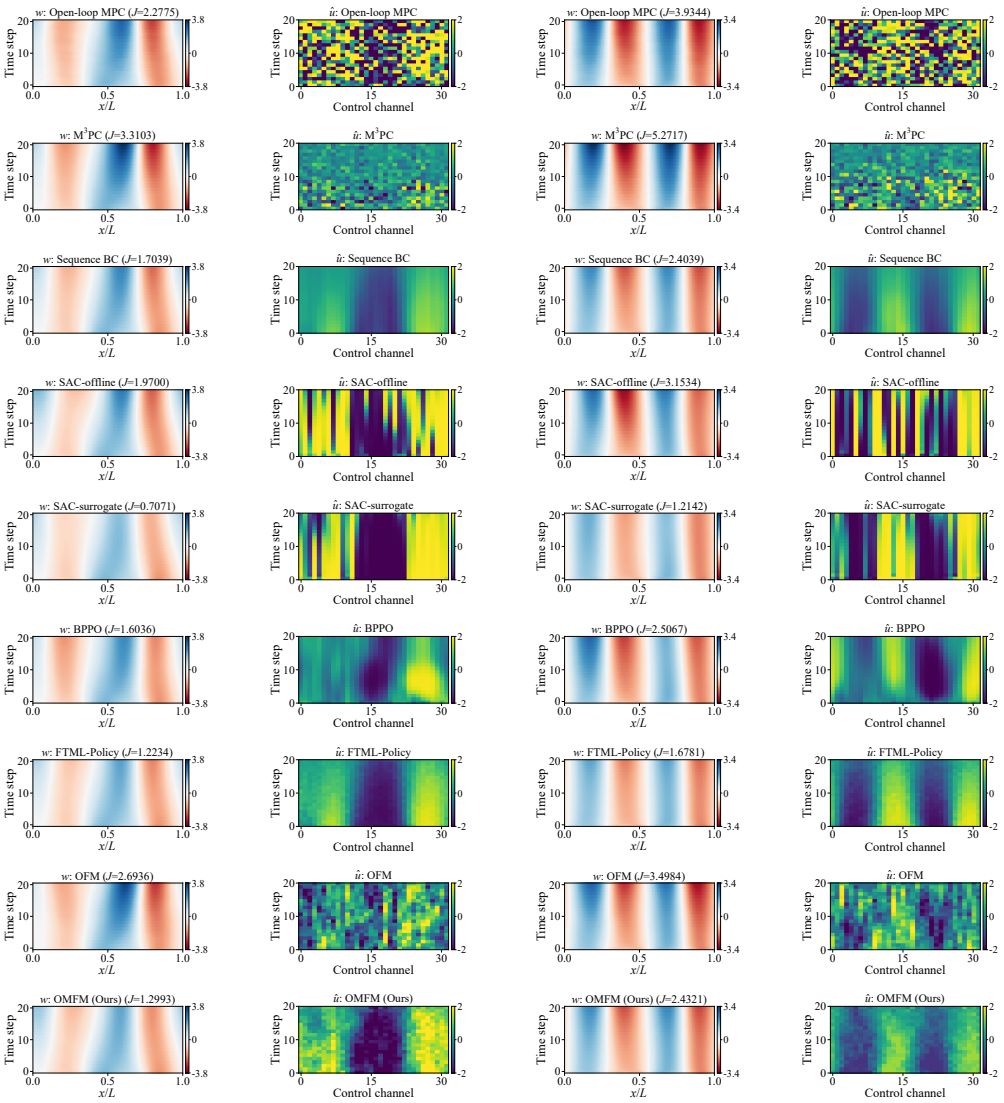


Figure 22: Comparison of OMFM and baseline methods on two additional OOD KS1D test cases, involving nonlinear spatiotemporal dynamics. Each pair of columns presents the state evolution and corresponding applied controls.

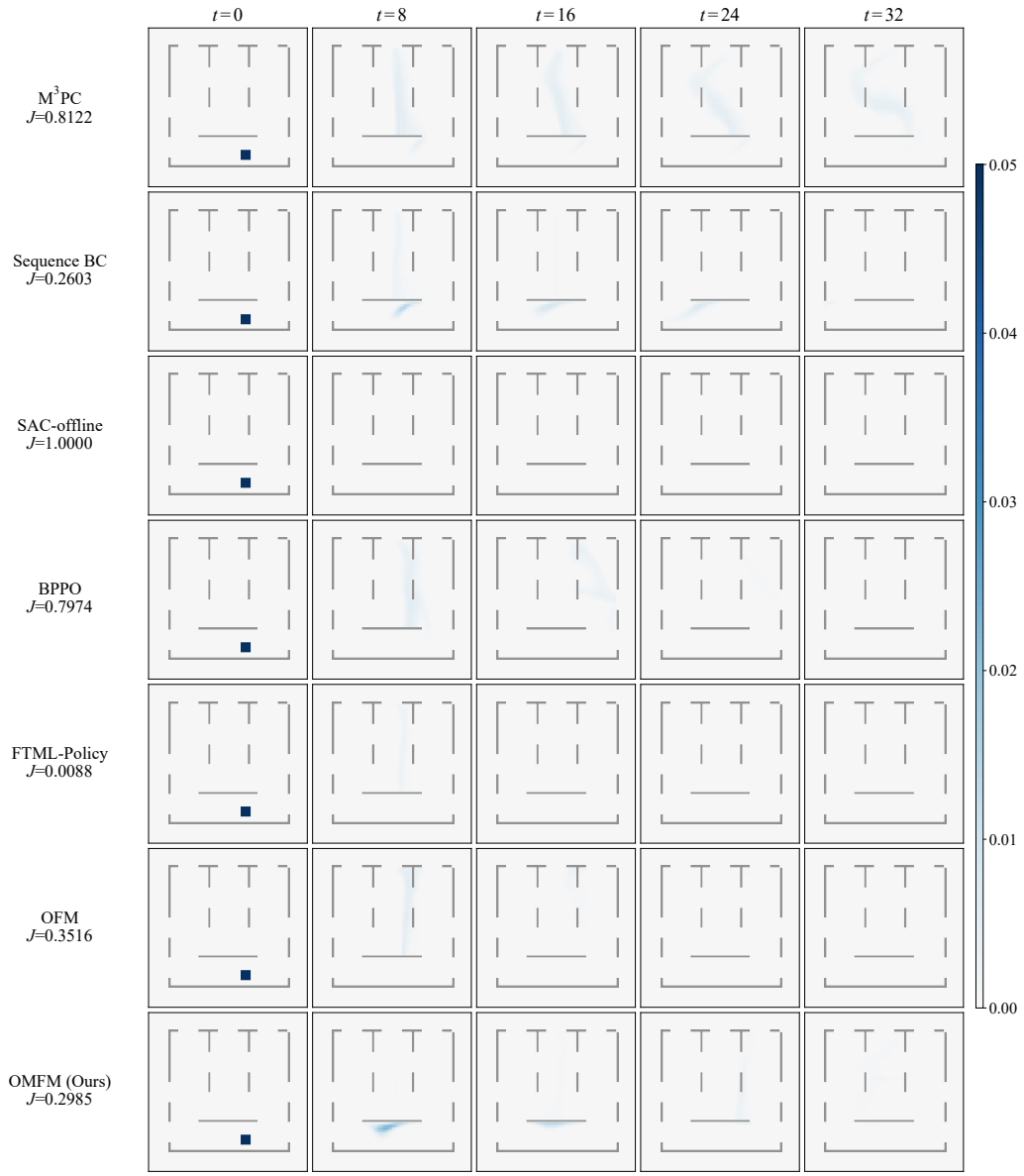


Figure 23: Comparison of OMFM and baseline methods on an additional OOD Smoke2D test case with a larger control region. Rows show density evolution around fixed obstacles while guiding smoke toward a target outlet.

E LIMITATIONS

We evaluate OMFM on finite-horizon control tasks with unknown dynamics and fixed parameters within each rollout. Our OOD evaluation establishes the scope of generalization under parameter shifts within each dynamics family, with governing mechanisms and environment parameters fixed within each rollout. Context-based adaptation uses observed trajectories to distinguish environment behavior at the resolution relevant to control. Candidate selection uses predicted costs to compare the finite set of controls proposed by the generator. The current evaluation uses open-loop execution over a prescribed horizon. The analysis characterizes conditional flow regression; the experiments assess realized finite-horizon control costs under the stated observation and actuation conditions. Future research will explore richer context acquisition, closed-loop extensions, and validation on real physical systems.

F BROADER IMPACT

OMFM provides a framework for controlling physical systems whose dynamics are unknown or vary across operating environments using accumulated trajectory experience. By accumulating transferable knowledge from sequentially available trajectories, it may reduce the need to train separate controllers for individual operating conditions. Predictor-assisted candidate selection also enables additional computational search without requiring physical trials for each candidate. These capabilities could benefit repeated experiments and industrial processes where data collection is costly and operating conditions change.