# Multi-Agent Differentiable Predictive Control for Zero-Shot PDE Scalability

**Pietro Zanotta*** **Dibakar Roy Sarkar*** **Honghui Zheng***

pzanott1@jhu.edu droysar1@jh.edu hzheng39@jh.edu

Johns Hopkins Whiting School of Engineering, Baltimore
3400 N Charles St, MD 21218, United States

**Abstract**

The control of large-scale systems governed by partial differential equations (PDEs) is often hindered by high dimensionality and the computational expense of real-time optimization. In this work, we present a Differentiable Predictive Control (DPC) framework that treats a PDE solver as a differentiable layer for policy optimization. We recast policy synthesis as an operator learning problem using the DeepONet (Reference [8]) architecture, mapping error fields to control actions across a continuous spatial domain. A key contribution is the demonstration of zero-shot scalability, where decentralized policies trained on a fixed swarm size $N$ generalize to unseen cardinalities $M$ without further tuning. We investigate a phenomenon of emergent self-normalization, conjectured to be a form of stigmergy. In this regime, the interaction between effort regularization and the physical field naturally discovers an $O(1/N)$ scaling of individual control intensities. Numerical experiments on the linear heat equation and nonlinear Fisher-KPP equation confirm that our decentralized approach maintains competitive performance while utilizing up to 76% fewer parameters than centralized benchmarks. These results suggest a robust foundation for deploying autonomous swarms in communication-constrained environments.

## 1 Introduction

The control of large-scale systems governed by partial differential equations (PDEs) remains a fundamental challenge in science and engineering, with applications ranging from environmental monitoring to swarm robotics. Traditional methods often struggle with the high dimensionality of the state space and the computational cost of real-time optimization. In this work, we leverage the Differentiable Predictive Control (DPC) paradigm [2], treating the PDE solver as a differentiable layer within a deep learning framework to compute exact sensitivity gradients for policy optimization.

The core contribution of our approach is the formulation of policy synthesis as an operator learning (References [6, 7]) problem using the DeepONet framework. In particular by mapping the current error field to control actions across a continuous spatial domain, our architecture inherently supports *zero-shot scalability*, allowing policies trained on a specific swarm cardinality $N$ to

*These authors contributed equally to this work.

generalize seamlessly to an unseen cardinality $M$ without further parameter tuning. We demonstrate this capability for both centralized and decentralized configurations, ensuring consistent forcing magnitude regardless of the number of active agents.

While centralized control provides a global performance benchmark, we focus on the more complex decentralized setting, where agents must operate with local-only sensing and a complete lack of inter-agent communication. This setup is particularly relevant for communication and memory-constrained environments, such as deep-sea exploration. Within this decentralized framework, we investigate the role of parameter sharing and conjecture the emergence of *self-normalization* among the agents as a form of *stigmergy* (an indirect form of coordination through the environment). In this regime, the optimization process naturally discovers an $O(1/N)$ scaling of individual control intensities, allowing the swarm to achieve collective stability and efficiency as an emergent property of the physical interaction with the field.

Through numerical experiments on the linear heat equation and the nonlinear Fisher-KPP equation, we empirically validate that our decentralized DPC framework not only maintains competitive performance with fewer parameters compared to the centralized counterpart with a small performance loss, but also robustly exhibits this cooperative efficiency gain as the swarm grows.

Schematically, our contributions are:

1. **Differentiable Operator Learning for Control**: we recast policy synthesis for PDE systems as an operator learning problem using the DeepONet framework. By treating the PDE solver as a differentiable layer through the Tesseract differentiable programming tool [9, 3], we compute exact sensitivity gradients for policy optimization.

2. **Decentralized Coordination without Communication**: We implement a decentralized control architecture where agents operate with local-only sensing and zero inter-agent communication and compare it to the centralized counterpart.

3. **Theoretical Gradient Consistency**: We provide a mathematical foundation (Theorem 1) ensuring that discrete policy gradients converge to the mean-field limit as the swarm size $N \to \infty$.

4. **Emergent Self-Normalization (Stigmergy)**: We conjecture and empirically investigate that the optimization process naturally discovers a $O(1/N)$ scaling of individual control intensities. This allows the swarm to achieve collective stability through physical interaction with the field rather than explicit coordination.

5. **Zero-Shot Scalability**: Our architecture inherently supports seamless generalization, allowing policies trained on a specific swarm cardinality to be deployed on unseen cardinalities without further parameter tuning.

6. **Empirical Validation and Efficiency**: Through numerical experiments on the linear heat equation and the nonlinear Fisher-KPP equation, we demonstrate that the decentralized framework achieves competitive performance with 48% fewer parameters than centralized benchmarks.

## 2 Problem formulation

### 2.1 System Dynamics

We consider a state field $z(x,t)$ defined over a bounded spatial domain $\Omega \subset \mathbb{R}^d$ and a temporal horizon $t \in [0,T]$. The evolution of the system is governed by a non-homogeneous nonlinear PDE:

$$\frac{\partial z(x,t)}{\partial t} = \mathcal{A}(z;\mu) + \mathcal{B}(x,t) \tag{1}$$

where:

- $\mathcal{A}(\cdot)$ is a nonlinear differential operator representing the intrinsic physics
- $\mu$ represents the physical parameters of the domain
- $\mathcal{B}(x,t)$ is the control forcing term generated by $N$ mobile actuators

### 2.2 Actuator Dynamics and Forcing

Each actuator $i \in \{1,\dots,N\}$ is characterized by its time-dependent position $\xi_i(t) \in \Omega$ and its control intensity $u_i(t) \in \mathbb{R}$. The actuators are modeled as first-order integrators:

$$\frac{d\xi_i(t)}{dt} = \mathbf{v}_i(t), \quad \xi_i(0) = \xi_{i,0} \tag{2}$$

and the total spatial forcing $\mathcal{B}(x,t)$ is the superposition of the individual actuator contributions, filtered through a spatial kernel (basis function) $b(x,\xi_i)$:

$$\mathcal{B}(x,t) = \sum_{i=1}^{N} b(x,\xi_i(t))u_i(t) \tag{3}$$

To ensure physical consistency and differentiability, we employ a Gaussian Kernel as the spatial mask:

$$b(x,\xi_i) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{|x-\xi_i|^2}{2\sigma^2}\right) \tag{4}$$

This formulation implies that as $\sigma \to 0$, the forcing approaches a sum of Dirac delta distributions, $\sum u_i \delta(x-\xi_i)$. In practice, $\sigma$ (a fixed parameter of the physical simulation) can be considered as the physical reach of the actuator.

### 2.3 The Control Objective

The control objective is to find an optimal control sequence $U^*(t) = \{u_i(t)\}_{i=1}^N$ and velocity sequence $V^*(t) = \{v_i(t)\}_{i=1}^N$ that minimizes a cost functional $\mathcal{J}$ involving a tracking cost $\mathcal{L}_{track}(z, z_{ref})$ a term $\mathcal{L}_{force}(u)$ discouraging from large energy consumption and $\mathcal{L}_{coll}(\xi)$ to prevent collisions between the actuators:

$$\min_{\mathbf{U},\mathbf{V}} \mathcal{J} = \mathbb{E}_{z_0 \sim \mathcal{D}} \left[ \int_0^T \left( \mathcal{L}_{track} + \lambda_u \mathcal{L}_{force} + \lambda_c \mathcal{L}_{coll} \right) dt \right]. \tag{5}$$

The optimization is subject to the following hard constraints on the actuator capabilities (saturation, kinematic limit and boundary containment):

$$|u_i(t)| \leq u_{max} \qquad |\mathbf{v}_i(t)| \leq v_{max} \qquad \xi_i(t) \in \Omega. \tag{6}$$

In order to find optimal control actions $U^*$ and $\mathbf{V}^*$ (here we drop the dependence on $t$ for ease of notation) for each time step, we rely on the Differentiable Predictive Control paradigm, where the control actions are the output of a neural architecture. Such a neural architecture is trained end-to-end, with the PDE solver being treated as a differentiable layer within the deep learning computational graph.

In order to make the PDE solver differentiable we wrap it in a Tesseract to allow seamless exposure of the sensitivity gradients of the future state with respect to the policy parameters in our Jax ecosystem.

Let $\mathbf{z}_k$ be the discretized state at time step $k$. The transition is defined by the differentiable operator $\Psi$:

$$\mathbf{z}_{k+1} = \Psi(\mathbf{z}_k, \mathbf{u}_k, \boldsymbol{\xi}_k; \Delta t) \tag{7}$$

which can be considered as the discretized version of Equation 1.

By chain-ruling through $\Psi$, we compute the sensitivity of the future state with respect to the control parameters $\theta$ through Backpropagation Through Time (BPTT). For a trajectory of length $K$, the total loss gradient is:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \sum_{k=1}^{K} \frac{\partial \mathcal{L}_k}{\partial \mathbf{z}_k} \left( \prod_{j=1}^{k-1} \frac{\partial \mathbf{z}_{j+1}}{\partial \mathbf{z}_j} \right) \frac{\partial \mathbf{z}_1}{\partial \theta} \tag{8}$$

### 2.4 Policy Parameterization

We utilize a neural architecture $\mathcal{G}_\theta$ to map observations to actions. Furthermore, since we are in a multi-agent setting, we enforce parameter sharing, i.e. every agent uses the same weights $\theta$. This means that our setup involves $N$ self-similar architectures (where $N$ is the cardinality of our swarm).

Algorithm 1 and Algorithm 2 provide an algorithmic formulation of this problem for the centralized and decentralized policy respectively.

## 3 Policy Learning as Operator Learning

The policy synthesis process is here recast as an operator learning problem. We define the control policy as a continuous mapping between function spaces:

$$\mathcal{G} : \mathcal{Z} \times \Omega \to \Phi, \quad \begin{bmatrix} \mathbf{v}(\xi, t) \\ u(\xi, t) \end{bmatrix} = \mathcal{G}(e(\cdot, t), \xi) \tag{9}$$

where $\Phi$ is the control field, $\mathcal{Z}$ is the global state of the system, $\Omega$ is the continuous spatial coordinate.

This problem description allows us to parametrize $\mathcal{G}_\theta$ as a neural operator that maps the current error field $e(x,t) = z(x,t) - z_{\text{ref}}(x,t)$ to a control action $a(\xi) = [u(\xi,t), \mathbf{v}(\xi,t)]^\top$ at any spatial query point $\xi \in \Omega$.

In particular we rely on the DeepONet framework, which provides an architecture specifically designed for learning nonlinear operators between infinite-dimensional function spaces.

**Algorithm 1** Multi-Agent Centralized DPC Training

**Input:** Initial state distribution $\mathcal{D}$, Horizon $T$, Number of actuators $N$, Reference $z_{ref}$
**Initialize:** Global Policy parameters $\Theta$, Learning rate $\eta$
**while** not converged **do**
  Sample initial condition $z_0 \sim \mathcal{D}$ and positions $\{\xi_i(0)\}_{i=1}^N$
  Initialize total loss $\mathcal{J} = 0$
  **for** $t = 0$ to $T$ **do**
    **Observe:** Construct global error field $e(\cdot, t) = z(\cdot, t) - z_{ref}(\cdot, t)$
    **Act:** Predict all actuator actions $U(t), V(t)$ simultaneously:
      $[U(t), V(t)] = \mathcal{G}_\Theta(e(\cdot, t), \{\xi_i\})$ // Global branch input
    **Step PDE:** Update global state $z(t + \Delta t)$ via differentiable solver $\Psi$
    **Step Actuators:** Update all positions $\xi_i(t + \Delta t) = \xi_i(t) + \Delta t \cdot v_i(t)$
    $\mathcal{J} \leftarrow \mathcal{J} + (\mathcal{L}_{track} + \lambda_u \mathcal{L}_{force} + \lambda_c \mathcal{L}_{coll})$
  **end for**
  **Update:** $\Theta \leftarrow \Theta - \eta \nabla_\Theta \mathcal{J}$ // Via BPTT through Tesseract
**end while**

**Algorithm 2** Multi-Agent Decentralized DPC Training (Parameter Sharing)

**Input:** Initial state distribution $\mathcal{D}$, Horizon $T$, Swarm cardinality $N$, Reference $z_{ref}$
**Initialize:** Shared Policy parameters $\theta$ (DeepONet branch $f$ and trunk $g$), Learning rate $\eta$
**while** not converged **do**
  Sample initial condition $z_0 \sim \mathcal{D}$ and positions $\{\xi_i(0)\}_{i=1}^N$
  Initialize total loss $\mathcal{J} = 0$
  **for** $t = 0$ to $T$ **do**
    Compute global error field: $e(\cdot, t) = z(\cdot, t) - z_{ref}(\cdot, t)$ // Proxy of having each agent sensing its own local patch
    **for** $i = 1$ to $N$ **do**
      **Observe:** Extract local patch $e_{patch,i}$ around agent position $\xi_i(t)$
      **Act:** Compute $u_i(t), v_i(t) = \mathcal{F}[f(e_{patch,i}), g(\hat{\xi}_i)]$ via DeepONet
    **end for**
    **Step PDE:** Update $z(t + \Delta t)$ using Tesseract differentiable solver $\Psi$:
      $z_{t+1} = \Psi(z_t, \{u_i\}, \{\xi_i\}; \Delta t)$
    **Step Actuators:** $\xi_i(t + \Delta t) = \xi_i(t) + \Delta t \cdot v_i(t)$
    $\mathcal{J} \leftarrow \mathcal{J} + (\mathcal{L}_{track} + \lambda_u \mathcal{L}_{force} + \lambda_c \mathcal{L}_{coll})$
  **end for**
  **Update:** $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{J}$ // Via BPTT through Tesseract
**end while**

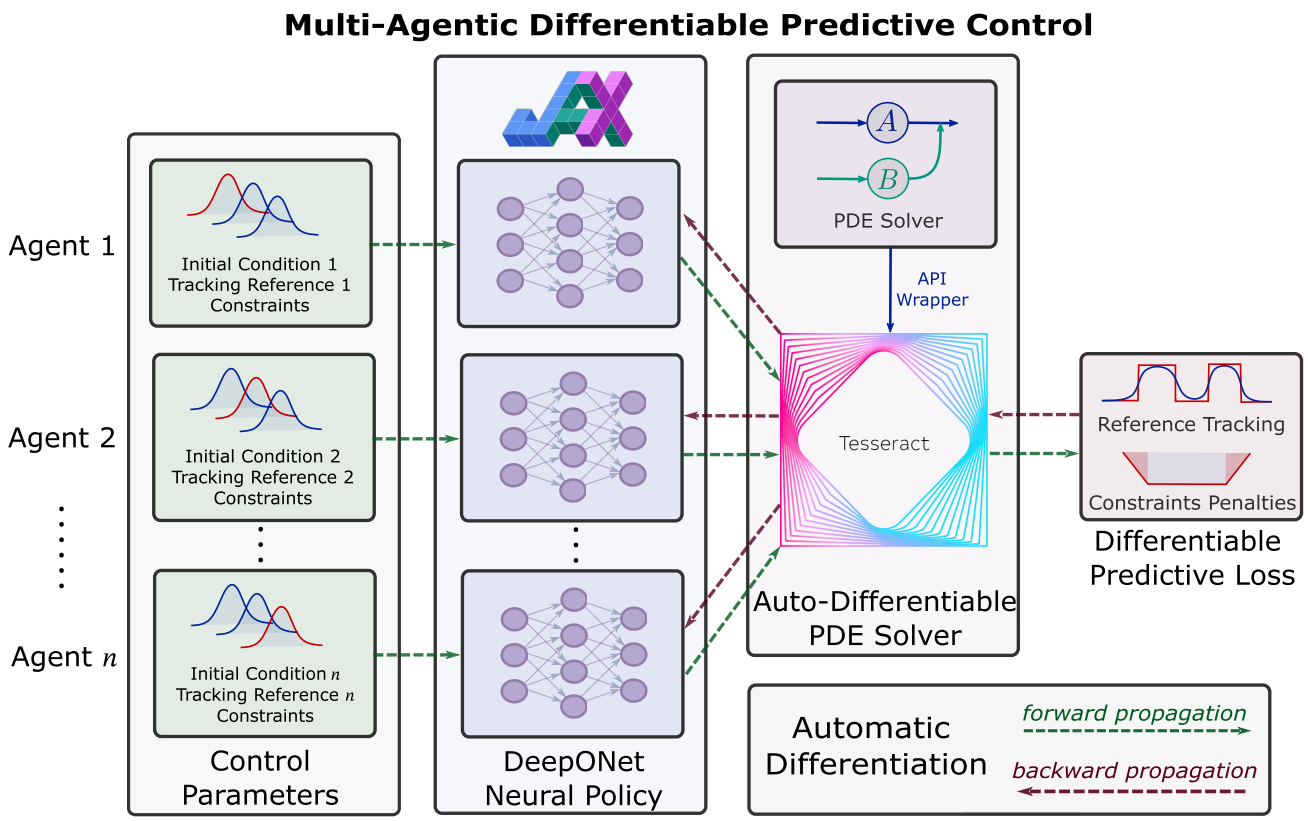



**Figure 1:** Schematic representation of Multi-Agentic DPC: each actuator is given its position, tracking reference and an observation of the target (this can be either local in the decentralized case or global in the centralized case). The neural policy outputs the $u_i$ and $\dot{\xi}_i$, Tesseract acts as the differentiable layer, evolving the simulation in time. The loss function involving tracking and the constraints is evaluated and differentiated through to update the neural policy weights.

It consists of two subnetworks: a branch net $\mathcal{B}r_\theta : \mathbb{R}^m \to \mathbb{R}^p$ that encodes the input function (in our case, the error field $e$) evaluated at a fixed set of $m$ sensor locations $\{x_1, \ldots, x_m\} \subset \Omega$, and a trunk net $\mathcal{T}r_\theta : \mathbb{R}^d \to \mathbb{R}^p$ that processes the query coordinates $\xi$ where the output is desired. The operator output is then obtained as:

$$\mathcal{G}_\theta(e)(\xi) = \mathcal{B}r_\theta\Big([e(x_1,t), \ldots, e(x_m,t)]\Big) \cdot \mathcal{T}r_\theta(\xi) \tag{10}$$

where $\cdot$ denotes the inner product in $\mathbb{R}^p$.

This architecture enables mesh-independent evaluation of the learned control policy at arbitrary spatial locations, making it particularly suited for PDE control problems (References [10, 4, 1]) where the control must be specified over continuous domains.

However, unlike standard DeepONets (e.g. Reference [5] for an application of DeepONet in Model Predictive Control), our operator output governs the physical trajectory of the "query points" themselves. The specific structure of the DeepONet is discussed in Section 4.

The next section explores the theoretical benefits of this formulation, including

- A theorem about the consistency of the policy gradients through a mean-field theory approach
- A conjecture about stigmergic behaviors of our implementation resulting in self-normalization of the $u_i$, allowing for zero-shot scalability: the ability of a policy trained on a swarm cardinality of $N$ to generalize to an unseen swarm cardinality $M$.

### 3.1 Mathematical Setup and Mean-Field Gradient

Let $H = L^2(\Omega)$ and $V = H_0^1(\Omega)$. We consider a linear operator $\mathcal{A} : V \to V^*$ that is strictly dissipative ($\langle \mathcal{A}z, z\rangle \leq -\alpha\|z\|_V^2$).

We define the parabolic regularity constant $C_T := \sup_{t\in[0,T]} \int_0^t \|S(t-s)\|_{\mathcal{L}(H,V)} ds < \infty$, where $\{S(t)\}_{t\geq 0}$ is the analytic $C_0$-semigroup generated by $\mathcal{A}$.

The normalized forcing for $N$ agents is defined using the empirical measure $\mu_N(t) = \frac{1}{N}\sum \delta_{\xi_i(t)}$ as:

$$\begin{aligned}\mathcal{B}_N(z)(x,t) &= \frac{1}{N}\sum_{i=1}^{N} b(x,\xi_i(t))\mathcal{G}_\theta(z(\cdot,t),\xi_i(t)) \\ &= \int_\Omega b(x,y)\mathcal{G}_\theta(z,y)\, d\mu_N(y) \end{aligned} \tag{11}$$

The mean-field gradient $\mathcal{D}_\theta\mathcal{J}_\infty$ is the sensitivity of the limit functional:

$$\mathcal{D}_\theta\mathcal{J}_\infty = \int_0^T \int_\Omega p_\infty(t,y)\nabla_\theta\mathcal{G}_\theta(z_\infty,y)\rho(y,t)\, dy\, dt \tag{12}$$

where $p_\infty$ satisfies the continuous adjoint equation $-\partial_t p_\infty + \mathcal{A}^* p_\infty = 2(z_\infty - z_{ref})$ with $p_\infty(T) = 0$ and $p_\infty|_{\partial\Omega} = 0$.

**Theorem 1** (Consistency of Discrete Policy Gradients)**.** *Let $z_0 \in H$ be independent of $N$. Assume the following hypotheses hold:*

***(H1) Measure Convergence:*** *$\mu_N(t) \rightharpoonup^* \rho(\cdot,t) \in L^\infty(\Omega)$ for almost every $t \in [0,T]$.*

***(H2) Policy Regularity:*** *$\mathcal{G}_\theta$ is $C^1$ in $(z,\theta)$ with Lipschitz constants $L_\pi, L_{\pi,\theta}$ such that $\|\mathcal{G}_\theta(z_1,\cdot) - \mathcal{G}_\theta(z_2,\cdot)\|_\infty \leq L_\pi\|z_1 - z_2\|_V$ and $\|\nabla_\theta\mathcal{G}_\theta(z_1,\cdot) - \nabla_\theta\mathcal{G}_\theta(z_2,\cdot)\|_\infty \leq L_{\pi,\theta}\|z_1 - z_2\|_V$.*

***(H3) Kernel Regularity:*** *$b \in L^2(\Omega\times\Omega)$ with $L_b := \sup_{y\in\Omega}\|b(\cdot,y)\|_H < \infty$ and $y \mapsto b(\cdot,y)$ is continuous in the $H$-topology.*

*If $L_\pi L_b C_T < 1$ is satisfied, then as $N \to \infty$: (i) $z_N \to z_\infty$ strongly in $L^2(0,T;H)$, (ii) $p_N \to p_\infty$ strongly in $L^2(0,T;H)$, and (iii) $\nabla_\theta\mathcal{J}_N \to \mathcal{D}_\theta\mathcal{J}_\infty$.*

*Proof Sketch:* We establish uniform bounds on the state $z_N$ and adjoint $p_N$ to invoke the Aubin-Lions lemma, ensuring strong convergence. We decompose the forcing error into a state-dependent Lipschitz term and a measure-dependent Portmanteau term. Finally, we pair strong convergence in the state and adjoint to prove gradient consistency.

The full proof is reported in Appendix A.

**Remark 1** (Nonlinear Extensions)**.** *The Fisher-KPP nonlinearity $f(z) = \rho z(1-z)$ used in our numerical experiments satisfies one-sided Lipschitz conditions that suggest analogous results hold, though a full proof is reserved for future work.*

### 3.2 Conjecture: Emergent Self-Normalization

While the theoretical guarantees of Theorem 1 rely on the normalized forcing $\mathcal{B}_N = \frac{1}{N}\sum u_i b$, our implementation uses the unnormalized forcing:

$$\mathcal{B}_N(x,t) = \sum_{i=1}^{N} b(x,\xi_i(t))u_i(t) \tag{13}$$

We conjecture that the optimization process induces an implicit normalization, causing the learned policy to exhibit the correct mean-field scaling.

**Conjecture 1** (Self-Normalization via Effort Regularization)**.** *Let $\theta_N^*$ denote the optimal policy parameters obtained by minimizing $\mathcal{J}_N$ with effort penalty $\lambda_u > 0$ and unnormalized forcing. As $N \to \infty$ with quasi-uniform agent distribution $\mu_N \rightharpoonup^* \rho$:*

1. ***(Intensity Scaling)*** *The learned control intensities satisfy:*

$$u_i^*(t) := \pi_{\theta_N^*}(z_N, \xi_i) = O(1/N) \quad \textit{uniformly in } t \tag{14}$$

2. ***(Forcing Consistency)*** *The total forcing remains bounded independently of $N$:*

$$\sup_N \|\mathcal{B}_N\|_{L^2(0,T;H)} \leq M < \infty \tag{15}$$

3. ***(Effort Decay)*** *The total control effort vanishes:*

$$\mathcal{L}_{force} = \sum_{i=1}^{N} |u_i^*(t)|^2 = O(1/N) \to 0 \tag{16}$$

**Heuristic Justification.** The self-normalization emerges as a dynamic equilibrium between physical tracking requirements and energetic parsimony. The optimization process seeks to satisfy two competing objectives:

- *Tracking Objective:* To achieve $\|z - z_{ref}\|^2 \leq \epsilon$, the environment must receive a total collective forcing $\mathcal{B}_N \approx F^*$, where $F^* = O(1)$ is the magnitude required by the underlying physics to shift the field state.
- *Effort Regularization:* The penalty $\lambda_u \sum_{i=1}^N u_i^2$ exerts a "pressure" on every individual actuator to minimize its own control intensity.

**The Scaling Logic**

If the policy produced individual intensities at a constant scale $u_i = O(1)$, the unnormalized total forcing would grow linearly with the swarm size:

$$\mathcal{B}_N = \sum_{i=1}^{N} u_i b(x, \xi_i) \approx N \cdot O(1) \cdot b = O(N) \tag{17}$$

In this regime, adding agents would cause the swarm to vastly overshoot the target, leading to a massive spike in tracking error. Conversely, if the policy discovers a scaling $u_i = O(1/N)$, the total forcing remains physically appropriate:

$$\mathcal{B}_N = \sum_{i=1}^{N} O(1/N) \cdot b = O(1) \tag{18}$$

This satisfies the tracking requirement while simultaneously minimizing the energy cost. Under this $1/N$ scaling, the total effort penalty becomes:

$$\mathcal{L}_{force} = \sum_{i=1}^{N} |u_i|^2 = N \cdot O(1/N^2) = O(1/N) \tag{19}$$

**Cooperative Efficiency Gain**

This result highlights a *"cooperative efficiency gain"*: as the swarm grows ($N \to \infty$), the total work required to control the system actually vanishes ($\mathcal{L}_{force} \to 0$). The swarm achieves collective stability by distributing the load across many participants such that each individual's contribution becomes infinitesimal.

**The Mechanism of Parameter Sharing**

The implementation of parameter sharing is the critical catalyst for this behavior. Because all agents share identical weights $\theta$, the policy $u_i = \mathcal{G}_\theta(z, \xi_i)$ cannot explicitly count the number of agents $N$ to adjust its gain. Instead, the scaling is discovered *stigmergically* through the state field $z$. Since $z$ is the superposition of all agent actions, it serves as a global ledger of the swarm's collective influence. During training, gradient descent tunes $\theta$ to respond to the error field; as the swarm density increases, the field reflects the cumulative reduction in error, naturally driving the shared policy toward the $O(1/N)$ equilibrium where tracking error and effort cost are balanced.

**Remark 2** (Recovery of Theoretical Guarantees)**.** *If Conjecture 1 holds, define the rescaled intensities $\tilde{u}_i := N u_i^* = O(1)$. The forcing can then be written as:*

$$\mathcal{B}_N = \sum_{i=1}^{N} u_i^* b = \frac{1}{N} \sum_{i=1}^{N} \tilde{u}_i \, b \tag{20}$$

*which recovers the normalized form assumed in Theorem 1. Thus, self-normalization provides the mechanism by which the closed-loop system implicitly satisfies the theorem's hypotheses, and the gradient consistency guarantees transfer to the unnormalized implementation.*

A rigorous proof of this conjecture is reserved for future work, while an empirical evidence of this is reported in Section 4.3.2.

**Remark 3** (Zero-Shot Scalability)**.** *Theorem 1 and Conjecture 1 together provide the theoretical foundation for zero-shot transfer across swarm sizes:*

1. ***Architectural:*** *The DeepONet structure defines $\mathcal{G}_\theta(z, \xi)$ for any $\xi \in \Omega$, enabling deployment with arbitrary agent configurations.*
2. ***Training:*** *Gradient consistency ensures that optimizing $\mathcal{J}_N$ with finite $N$ yields policies that approximate the mean-field optimum $\theta_\infty^*$.*
3. ***Deployment:*** *Self-normalization ensures that the learned policy produces forcing of consistent magnitude regardless of $N$, preventing over- or under-actuation.*

*A complete transfer bound (quantifying suboptimality when deploying $\theta_N^*$ with $M \neq N$ agents) additionally requires stability of the minimizers $\theta_N^* \to \theta_\infty^*$, which we leave to future work. The experiments in Section 4 empirically validate successful transfer.*

# 4 Numerical Experiments

For 1d experiments: Our policy $\mathcal{G}_\theta$ is structurally equivalent to a DeepONet with a modified fusion layer. It consists of two primary components:

- The Branch net $f$ processes the global or local state $e(x,t)$. For a set of $P$ sensor measurements $\{e(x_j,t)\}_{j=1}^{P}$, the branch produces a latent descriptor $\mathbf{c} \in \mathbb{R}^p$:

$$\mathbf{c} = f(e(x_1,t), \ldots, e(x_P,t)). \tag{21}$$

  Moreover, in our multi-agent setup, this branch acts on a local patch centered at $\xi_i$, effectively acting as a convolutional kernel that extracts local features of the PDE state with no communication between the actuators.

- The Trunk net $g$ takes the actuator's spatial coordinates $\xi \in \Omega$ as input. To capture high-frequency spatial variations in the control law, we employ Fourier Feature Encoding $\hat{\xi}$:

$$\hat{\xi} = [\cos(2\pi\mathbf{W}\xi), \sin(2\pi\mathbf{W}\xi)]^\top. \tag{22}$$

  The trunk output $g(\hat{\xi})$ ensures that the control law is defined over the continuous domain $\Omega$, not just at discretized grid points.

- In a standard DeepONet, the branch and trunk are combined via a dot product. In our architecture, we use an MLP fusion $\mathcal{F}$ to allow for non-linear interactions between the physical context and the actuator position:

$$\left[u_i(t)\ \mathbf{v}_i(t)\right] = \mathcal{F}\left[f(\text{patch}_i), g(\hat{\xi}_i)\right] \tag{23}$$

Identical copies of this policy are then deployed on each actuator.
The hyperparameters and the simulation details are reported in Appendix B.

### 4.1 Experiment 1: 1D Heat Equation Control

#### 4.1.1 Problem Setup

We validate our approach on the 1D heat equation with Dirichlet boundary conditions:

$$\frac{\partial z}{\partial t} = \nu \frac{\partial^2 z}{\partial x^2} + B(x,t), \quad x \in [0,1] \tag{24}$$

Control forcing is applied through $N = 8$ mobile Gaussian actuators (3) with $|u_i| \le u_{\max}$, $|v_i| \le v_{\max}$. The objective minimizes tracking error to a target profile $z_{\text{ref}}(x)$ while penalizing control effort, boundary violations, and inter-agent collisions:

$$\min_{\theta} \mathbb{E}\left[\int_0^T \left(\|z - z_{\text{ref}}\|_2^2 + \lambda_u \sum u_i^2 + \ldots\right) dt\right] \tag{25}$$

The initial conditions and reference states are sampled from a Gaussian Random Field (GRF) with length scale $\ell_{\text{init}} = 0.2$ and $\ell_{\text{target}} = 0.4$ respectively. GRF generation applies bridge correction to satisfy zero boundary conditions. We train and deploy with 8 agents.

More information about the trainig setup can be found in Appendix B

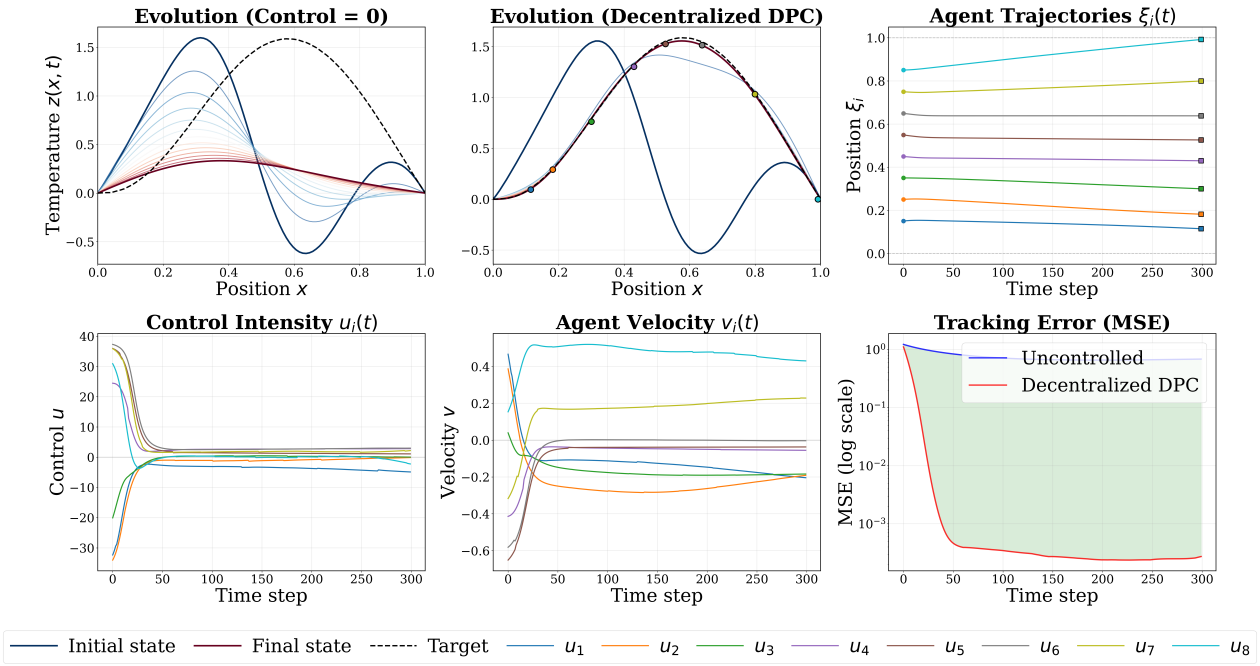



**Figure 2:** DPC statistics for the decentralized policy for Heat equation

### 4.1.2 Results and Analysis

The centralized controller achieves tracking loss below 5.2e-3 while the decentralized variant converges to approximately 6.4e-3 despite local-only sensing. Both controllers successfully learn diffusion-aware strategies, positioning actuators ahead of target regions to account for heat spreading. It is also relevant that control effort concentrates in early timesteps with intensity correlating strongly with local error magnitudes, and no collisions occur despite soft penalty formulation as shown in Figure 2 and Figure 3.

The decentralized architecture provides compelling practical advantages as the 48% fewer parameters translates into a modest 23% decrease in the tracking error. Such an advantage, coupled communication-free operation make our decentralized setup an interesting candidate for deep-sea swarm exploration and subterrenean environments or in any scenario where communication between agents is either too expensive or impossible. Both architectures train in approximately 1 minute on a for 500 epochs, with the efficiency gains realized during deployment. We believe that in higher dimensional PDEs we can expect also a faster training from the decentralized policy due to the smaller number of parameters of the network and this is left for future works.

**Table 1:** Heat Equation: Architecture and Performance Comparison

| Metric | Centralized | Decentralized |
|---|---|---|
| Branch Input Dim | 200 | 40 |
| Total Parameters | 21,794 | 11,298 |
| Final Tracking Loss | 5.2e-3 | 6.4e-3 |
| Scalability | Zero-shot | Zero-shot |
| Communication | Global | None |
| Training Time | ∼1 min | ∼1 min |

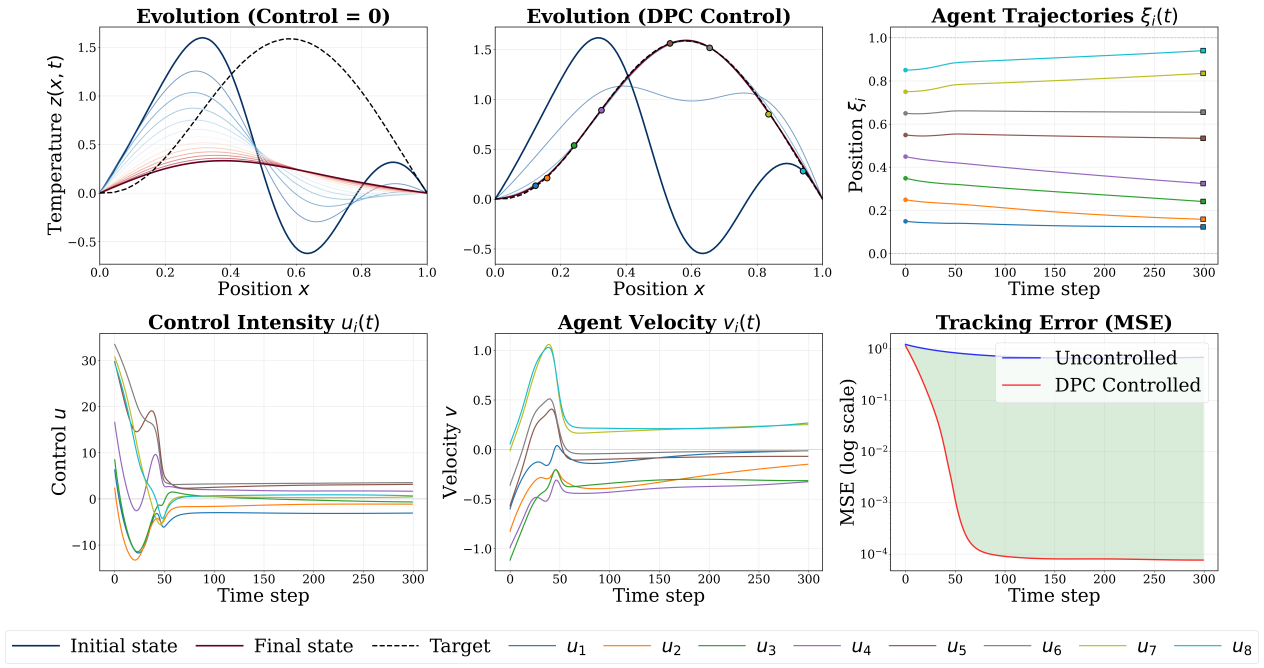



**Figure 3:** DPC statistics for the centralized policy for Heat equation

## 4.2 Experiment 2: Fisher-KPP Reaction-Diffusion Control

### 4.2.1 Problem Setup

To validate generalization beyond linear dynamics, we consider the Fisher-Kolmogorov-Petrovsky-Piskunov (FKPP) equation modeling population dynamics, chemical fronts, and biological invasions. The 1D Fisher-KPP equation introduces nonlinear reaction dynamics through a logistic growth term:

$$\frac{\partial z}{\partial t} = \nu \frac{\partial^2 z}{\partial x^2} + \rho z(1 - z) + B(x, t) \tag{26}$$

with Dirichlet boundaries $z(0, t) = z(1, t) = 0$. The logistic term $\rho z(1-z)$ creates bistable behavior with unstable extinction equilibrium at $z = 0$ and stable carrying capacity at $z = 1$. This fundamentally differs from heat diffusion: controllers must work against natural growth when suppressing population and can leverage intrinsic growth when enhancing it. Control forcing employs the same Gaussian actuator model, where $u_i < 0$ represents suppression and $u_i > 0$ represents enhancement. The objective function remains identical to the heat equation case.

### 4.2.2 Implementation Details

Data generation requires non-negative, bounded profiles satisfying boundary conditions. We sample raw Gaussian Random Fields $f(x) \sim \mathcal{GP}(0, K_\ell)$, apply exponential transform and boundary envelope via $\sin^2(\pi x)$, then normalize to $[0, 1]$:

$$z(x) = \frac{\exp(f(x)) \cdot \sin^2(\pi x)}{\max_y [\exp(f(y)) \cdot \sin^2(\pi y)]} \tag{27}$$

This ensures physical consistency with carrying capacity constraints while maintaining smooth, biologically plausible profiles.

The policy architectures remain identical to the heat equation experiment with all the training details highlighted in Appendix B.

### 4.2.3 Results and Analysis

The Fisher-KPP system presents substantially increased control complexity compared to linear heat diffusion. The competing dynamics between natural growth and control objectives, combined with sharper spatial features from reduced diffusion and bounded state space saturation effects, elevate tracking difficulty. To tackle this complexity, we train and deploy 20 actuators, resulting in the centralized controller achieving a final tracking loss of approximately 7e-3, while the decentralized variant converges to 8e-3.

The learned policies exhibit several distinctive behaviors adapted to the nonlinear system. Agents demonstrate asymmetric control strategies, applying stronger suppression intensities than enhancement since reducing population requires overcoming natural growth toward carrying capacity. Spatial positioning shifts toward front-tracking behavior, with actuators concentrating at population front boundaries where small control inputs achieve maximum effect due to the steep gradients. Control effort remains higher throughout trajectories compared to purely diffusive systems, as the reaction term continuously drives the state away from certain target configurations. Despite these challenges, the decentralized controller maintains reasonable performance through local error gradient sensing, demonstrating that stigmergic coordination mechanisms extend to nonlinear reaction-diffusion systems as shown in Figure 4 and Figure 5.

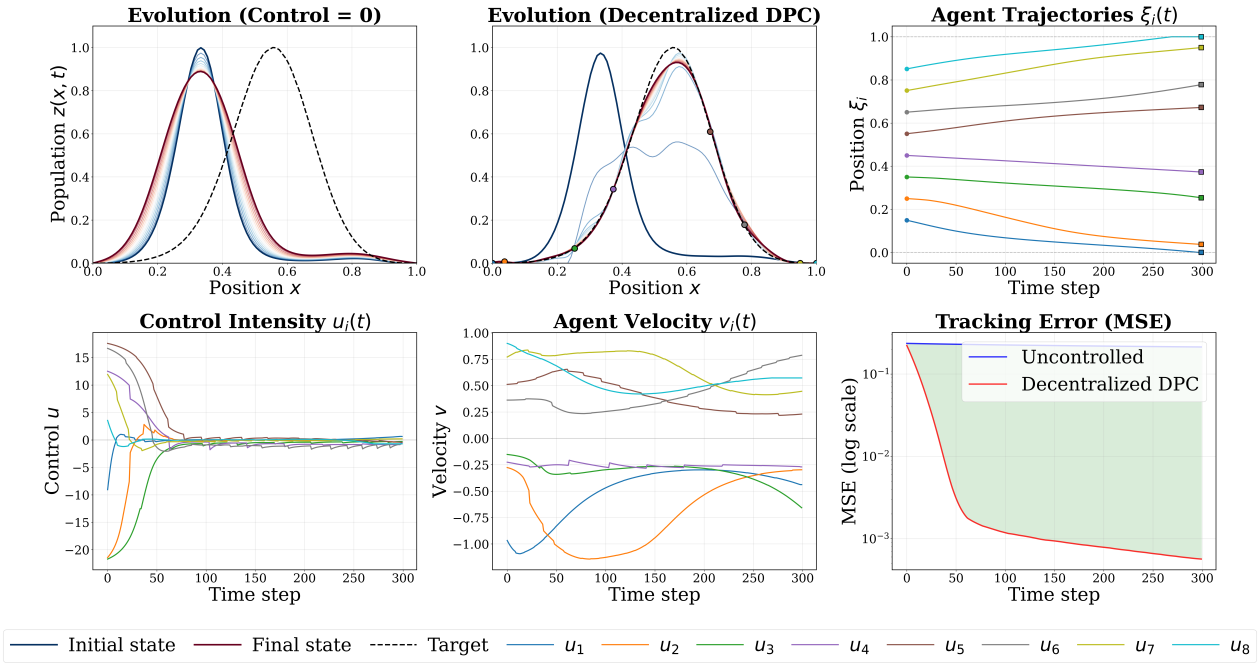



**Figure 4:** DPC statistics for the decentralized policy for FKPP equation

These results establish that the differentiable predictive control framework generalizes beyond linear PDEs to handle nonlinear dynamics with state constraints. Gradients propagate correctly through the logistic reaction term, clipping operations are handled via straight-through gradient estimation, and the policy learns to balance fast reaction dynamics with slow diffusion. This suggests broader applicability to reaction-diffusion systems including Allen-Cahn phase transitions, Gray-Scott pattern formation, and chemotaxis models.

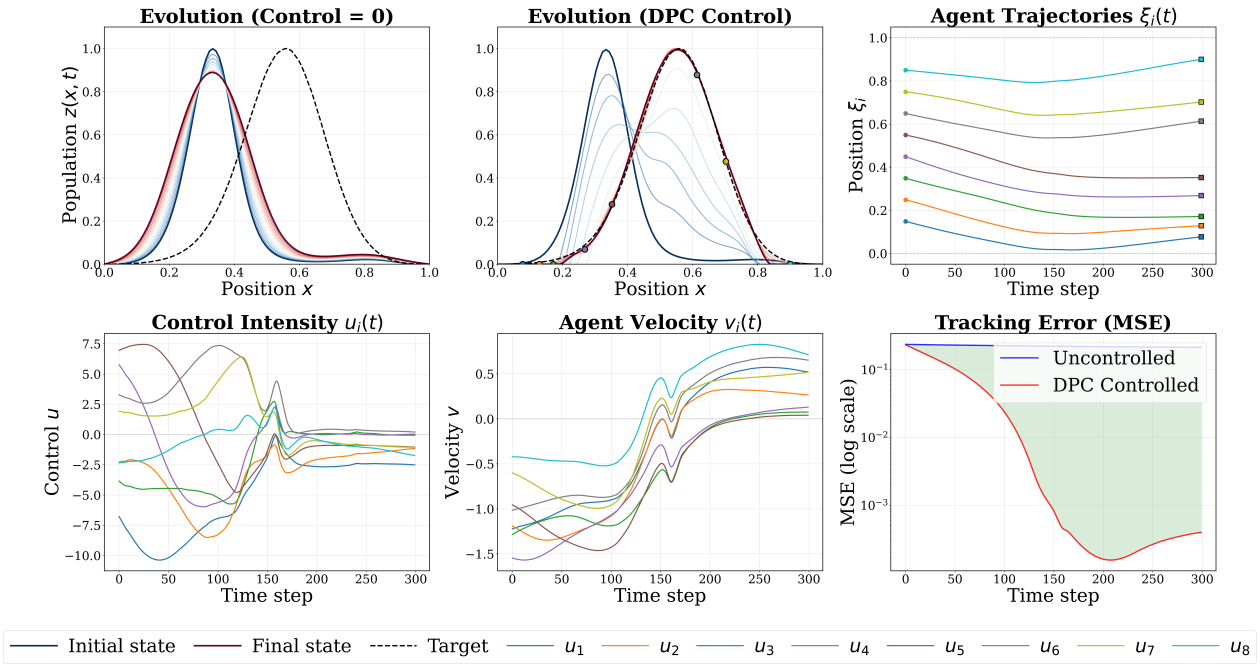



**Figure 5:** DPC statistics for the centralized policy for FKPP equation

**Table 2:** FKPP Equation: Architecture and Performance Comparison

| Metric | Centralized | Decentralized |
|---|---|---|
| Branch Input Dim | 200 | 40 |
| Total Parameters | 21,794 | 11,298 |
| Final Tracking Loss | 7.0e-3 | 8.3e-3 |
| Scalability | Zero-shot | Zero-shot |
| Communication | Global | None |
| Training Time | ∼3 min | ∼3 min |

### 4.3 Experiment 3: 2D Heat Equation Control

In this last experiment, we use a setup similar to the one discussed in Section 4.1. Control forcing is now applied through $N = 16$ mobile Gaussian actuators of the form in Equation (3) instead of 8 to account for the complexity of a 2d problem.

The initial conditions and reference states are sampled from a 2d Gaussian Random Field (GRF) with length scale $\ell_{\text{init}} = 0.25$, $\ell_{\text{target}} = 0.4$.

More information about the training setup can be found in Appendix B.

#### 4.3.1 Results and Analysis

The centralized policy results in a tracking loss of 7.8e-3 while the decentralized variant converges to approximately 9.0e-3. From a qualitative perspective, both controllers successfully learn diffusion-aware strategies, as evident in Figure 7 and Figure 6.

This 2D scenario is where the decentralized architecture provides a real advantage, with 76% fewer parameters compared to the centralized model with a performance degradation of roughly 13%, validating the relevance of the proposed methodology. More details about this experiments

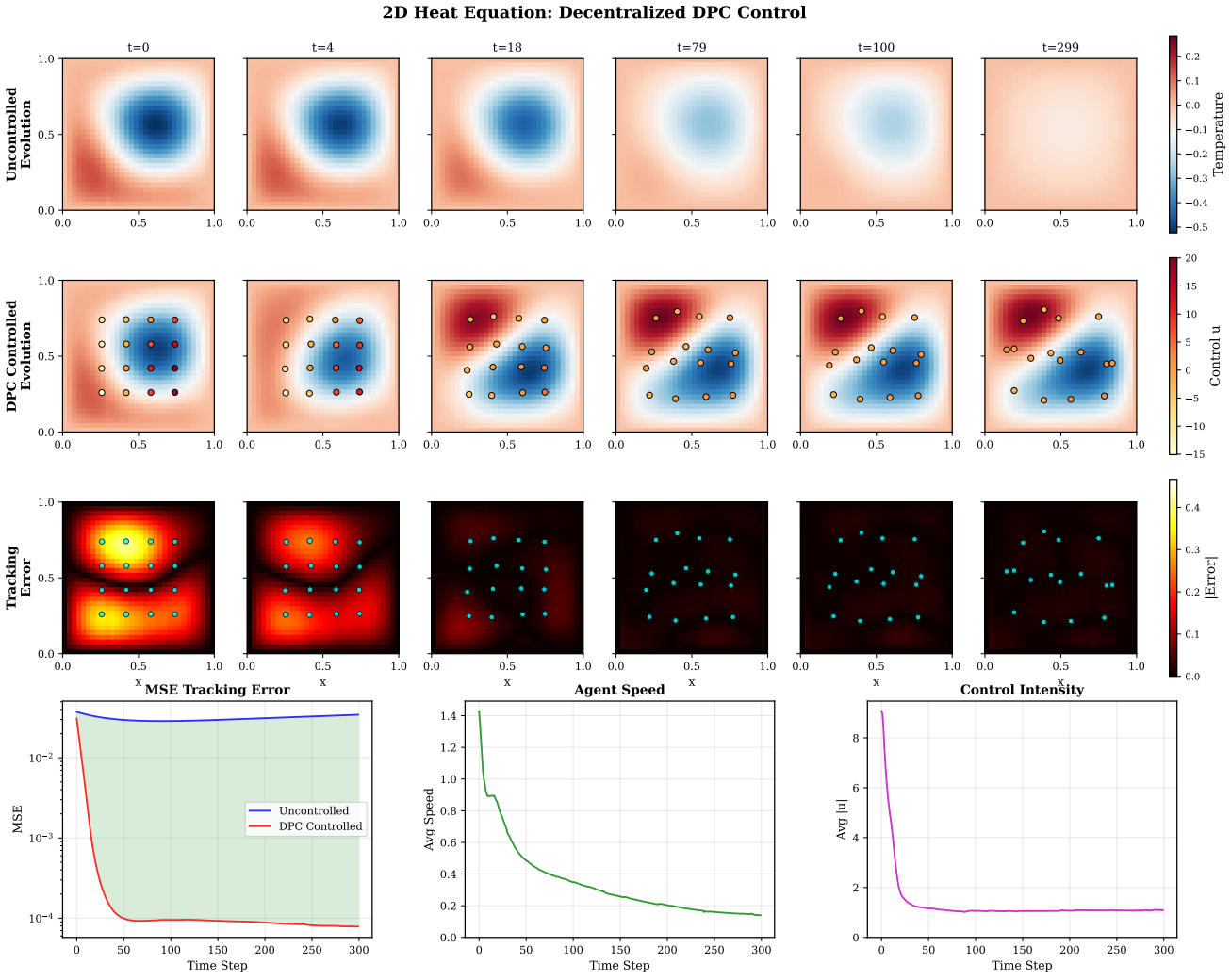



**Figure 6:** DPC statistics for the decentralized policy for 2D Heat equation

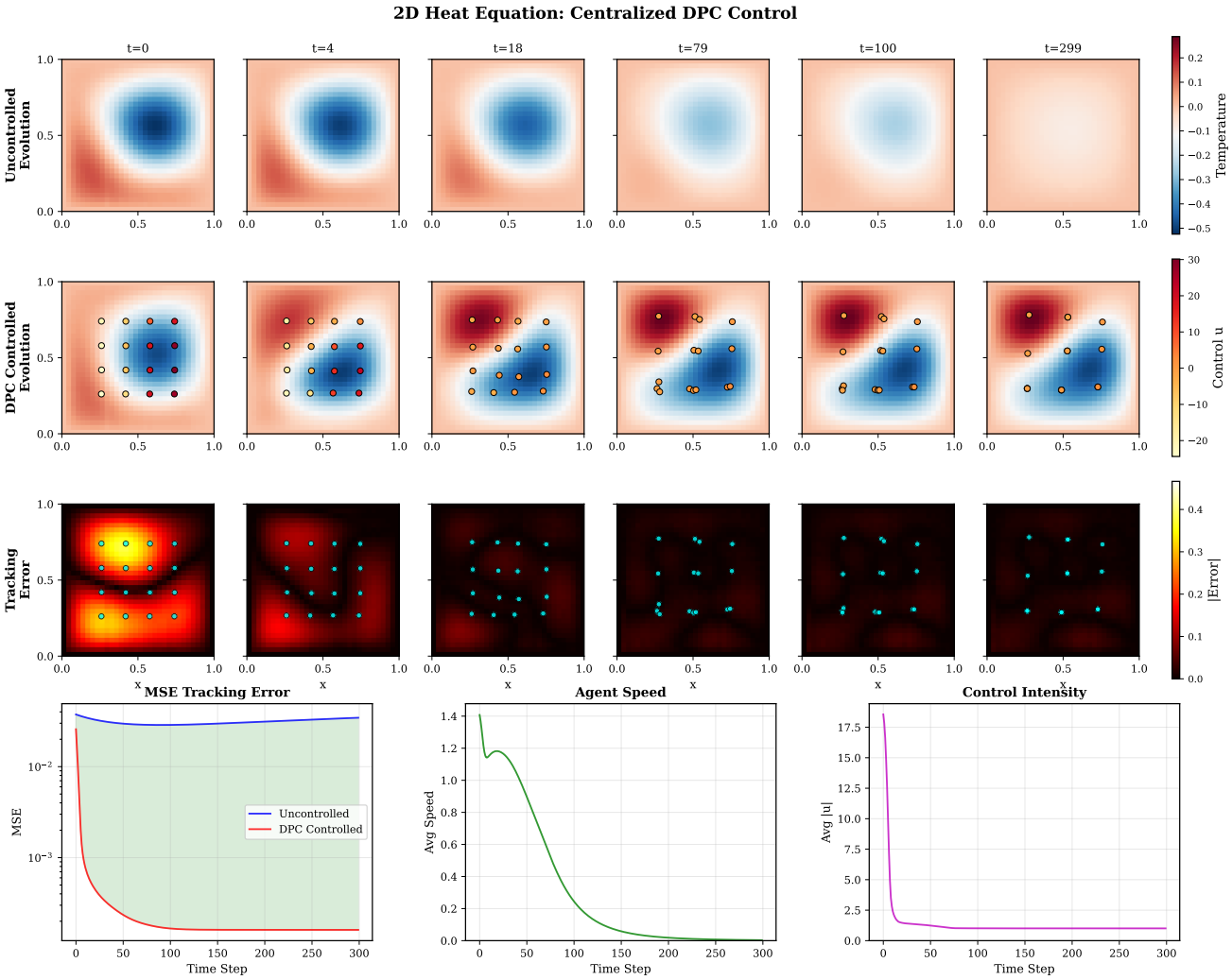



**Figure 7:** DPC statistics for the centralized policy for 2d Heat equation

are reported in 3.

### 4.3.2 Empirical Validation of Emergent Self-Normalization

This section provides an empirical validation of Conjecture 1 within the nonlinear Fisher-KPP (FKPP) framework. We focus on three critical phenomena: the $O(1/N)$ scaling of individual control intensities $u_i$, the $O(1)$ stability of the collective forcing $\|\mathcal{B}\|$, and the resulting zero-shot scalability. While these properties are inherent to centralized models, they are essential for decentralized policies; in the absence of global communication, the system cannot explicitly coordinate actuator outputs, making emergent self-normalization the primary mechanism for swarm stability.

To evaluate these properties, we utilize a decentralized policy trained on a fixed swarm cardinality of $N = 20$ (indicated by the red vertical line in Figure 8) and deploy it across configurations ranging from $N = 10$ to $N = 60$ without further parameter tuning.

- **Zero-Shot Scalability and Actuation Regimes:** As illustrated by the blue curve in Figure 8, the Tracking MSE demonstrates successful zero-shot transfer. For $N < 30$, the system remains underactuated, as the sparse controller density is insufficient to fully suppress

**Table 3:** 2D Heat Equation: Architecture and Performance Comparison

| Metric | Centralized | Decentralized |
|---|---|---|
| Branch Input Dim | 1024 | 144 |
| Total Parameters | 2,116,003 | 158,531 |
| Final Tracking Loss | 7.8e-3 | 9.0e-3 |
| Scalability | Zero-shot | Zero-shot |
| Communication | Global | None |
| Training Time (500 epochs) | ∼4 min | ∼3 min |

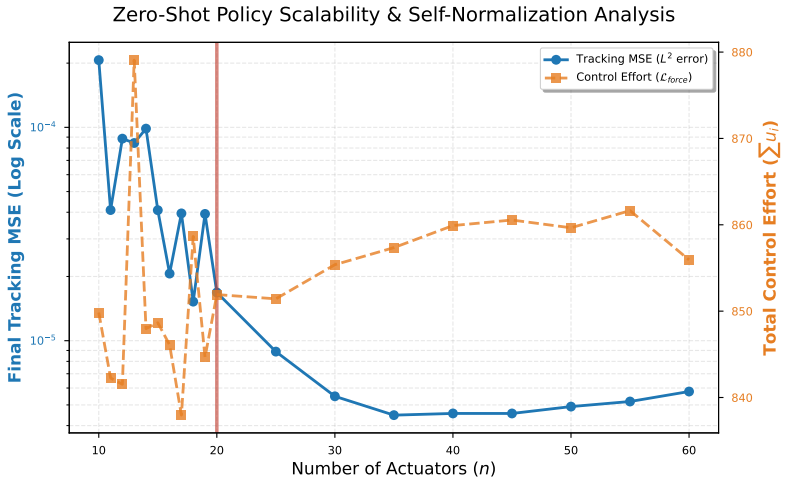


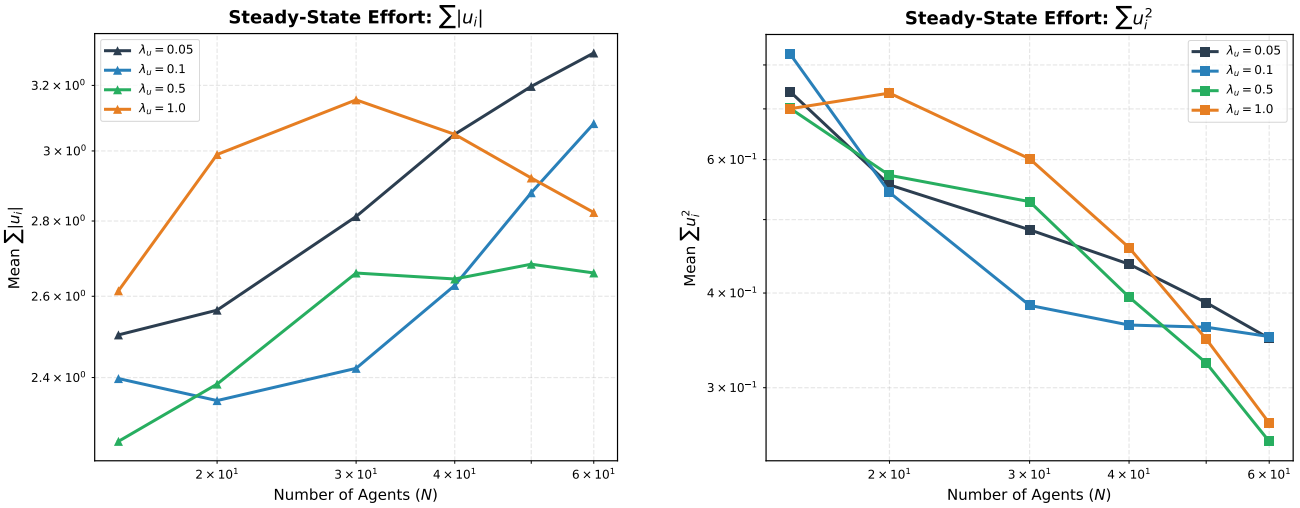

**Figure 8:** Zero-shot scalability and self-normalization analysis for the decentralized FKPP policy. The blue curve tracks the tracking MSE, while the orange curve monitors total control effort.

or enhance the population fronts to reach the target state. Crucially, as $N$ increases beyond the training cardinality, the error stabilizes rather than diverges, confirming the policy's robustness to varying swarm scales.

- **Bounded Collective Forcing:** The total control effort ($\sum u_i$), represented by the orange curve in Figure 8, remains bounded within a narrow range (approximately 835 to 880) despite a sixfold increase in agent count. This lack of energy "blowup" provides strong evidence for the hypothesized $u_i = O(1/N)$ intensity scaling.

- **Cooperative Efficiency and Stigmergy:** Figures 9a and 9b validate the "cooperative efficiency gain" mechanism. We hypothesize that stigmergic coordination emerges primarily in steady-state; thus, we analyze the final 70% of the simulation horizon. In the transient phase, the optimizer trades efficiency for rapid tracking error reduction; as the error gradient diminishes, efficiency becomes the dominant driver, triggering self-normalization.

  Specifically, Figure 9a demonstrates that across various effort penalties, the absolute effort sum $\sum |u_i|$ remains stable—varying by only $\approx 20\%$ even as the number of actuators triples. More decisively, Figure 9b reveals a consistent decrease in squared effort $\sum u_i^2$ as $N$ increases. This confirms a $O(1/N)$ decay in individual workloads as swarm density increases, proving that agents perform progressively less work as the swarm grows, leveraging collective density to achieve the global control objective.




**(a)** Steady-state absolute effort $\sum |u_i|$

**(b)** Steady-state squared effort $\sum u_i^2$

**Figure 9:** Scaling of effort metrics across varying agent counts. Left: Absolute effort. Right: Squared effort demonstrating $O(1/N)$ decay.

These findings establish the differentiable predictive control framework as a robust and scalable foundation for deploying autonomous swarms in communication-constrained environments, such as, for instance, deep-sea exploration.

## 5 Conclusion and Future Work

This paper demonstrates that the synthesis of control policies for PDE-governed systems can be effectively framed as an operator learning problem through the DeepONet framework. By integrating differentiable physics with neural operators, we have established a control paradigm that is inherently scalable and continuous.

Our findings provide both theoretical and empirical support for several key advancements:

1. **Decentralized Efficiency**: We show that agents operating with only local sensing and no inter-agent communication can achieve collective stability. The decentralized variant achieved tracking losses (approx. 6.4e-3 for Heat and 8.3e-3 for FKPP) that are closely comparable to centralized benchmarks while significantly reducing model complexity.

2. **The Power of Stigmergy**: Our results empirically validate the conjecture of self-normalization. As swarms grow, agents leverage a cooperative efficiency gain, where individual workloads decay at a rate of $O(1/N)$, while the collective forcing remains stable and bounded.

3. **Zero-Shot Scalability**: The architecture demonstrates remarkable robustness, allowing a policy trained on 20 actuators to be deployed across configurations ranging from 10 to 60 agents without performance divergence

Future work will focus on extending these results to a wider class of PDEs, testing the limitation of this approach and providing a rigorous proof for the self-normalization conjecture.

## References


[1] Luke Bhan, Yuanyuan Shi, and Miroslav Krstic. Neural operators for bypassing gain and control computations in pde backstepping. *IEEE Transactions on Automatic Control*, 69(8):5310–5325, Dec 2023.

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

[3] Daniel Häfner and Alexander Lavin. Tesseract core: Universal, autodiff-native software components for simulation intelligence. *Journal of Open Source Software*, 10(111):8385, 2025.

[4] Philipp Holl, Vladlen Koltun, and Nils Thuerey. Learning to control pdes with differentiable physics, 2020.

[5] Lazar Jong, Shukla. Deep operator neural network model predictive control, 2025.

[6] Nikola Kovachki, Nvidia Li, Kamyar Azizzadenesheli, Nvidia Bhattacharya, Andrew Stuart, Anima Anandkumar, Lorenzo Rosasco, Nikola 2023, Zongyi Kovachki, Burigede Li, Kamyar Liu, Kaushik Azizzadenesheli, Andrew Bhattacharya, and Anima Stuart. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24:1–97, 2023.

[7] Nikola B Kovachki, Samuel Lanthaler, and Andrew M Stuart. Operator learning: Algorithms and analysis, 2024.

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

[9] Andrin Rehmann, Dion Häfner, Alessandro Angioi, Yongquan Qu, and Andrei Paleyes. Pipeline-level differentiable programming for the real world. *Proceedings of the Python in Science Conference*, page 278–285, Jul 2025.

[10] Dibakar Roy Sarkar, Ján Drgoňa, and Somdatta Goswami. Learning to control pdes with differentiable predictive control and time-integrated neural operators, 2025.

# A Full Proofs

*Proof.* **Step 1: Uniform Bounds and Strong State Convergence.** The normalization ensures $\|\mathcal{B}_N(z)\|_H \leq L_b\|\mathcal{G}_\theta\|_\infty$. Under $L_\pi L_b C_T < 1$, the Picard map is a contraction, yielding a unique $z_N$. Energy estimates and Gronwall's lemma yield $\|z_N\|^2_{L^\infty(H)} + \|z_N\|^2_{L^2(V)} \leq C$ uniformly. By the **Aubin-Lions Lemma**, $\{z_N\}$ is relatively compact in $L^2(0,T;H)$, yielding a strongly convergent subsequence $z_N \to z_\infty$.

**Step 2: Forcing Convergence.** We decompose the forcing error:

$$\begin{aligned}\|\mathcal{B}_N(z_N) - \mathcal{B}_\infty(z_\infty)\|_H \leq &\|\mathcal{B}_N(z_N) - \mathcal{B}_N(z_\infty)\|_H \\ &+ \|\mathcal{B}_N(z_\infty) - \mathcal{B}_\infty(z_\infty)\|_H \end{aligned} \tag{28}$$

The first term $\leq L_b L_\pi \|z_N - z_\infty\|_V \to 0$. For the second term, since $y \mapsto b(\cdot, y)\mathcal{G}_\theta(z_\infty, y)$ is continuous and bounded by (H2-H3), the **Portmanteau Theorem** for weak-* convergence of $\mu_N$ ensures the integral converges to $\mathcal{B}_\infty(z_\infty)$.

**Step 3: Adjoint Stability and Gradient Consistency.** The adjoint $p_N$ satisfies $-\partial_t p_N + \mathcal{A}^* p_N = 2(z_N - z_{ref})$. Since $z_N \to z_\infty$ strongly, the source term is bounded in $L^2(H)$, implying $\{\partial_t p_N\}$ is bounded in $L^2(V^*)$. A second application of **Aubin-Lions** yields $p_N \to p_\infty$ strongly in $L^2(0,T;H)$. The gradient $\nabla_\theta \mathcal{J}_N = \int_0^T \langle p_N, \nabla_\theta \mathcal{B}_N \rangle dt$ converges to $\mathcal{D}_\theta \mathcal{J}_\infty$ as both components converge strongly in their respective spaces. □

# B Experimental Details

This section details the architectural configurations, training hyperparameters, and numerical discretization schemes used for the Heat and FKPP equation experiments.

## B.1 Architecture Comparison

The architectures are designed as DeepONet variants where a branch network processes the error field and a trunk network processes the spatial coordinates. The primary distinction between the centralized and decentralized approaches lies in the input dimensionality of the branch network.

**Table 4:** Architectural specifications for the controller models.

| **Feature** | **Centralized** | **Decentralized** |
|---|---|---|
| Input Type | Global $(e, \nabla e)$ | Local patch |
| Input Dimension | $\mathbb{R}^{200}$ | $\mathbb{R}^{40}$ |
| Branch Network | [64, 64] MLP | [64, 64] MLP |
| Trunk Network | [32, 32] MLP | [32, 32] MLP |
| Total Parameters | 21,794 | 11,298 |
| Limits | $u_{\max} = 40.0$ | $u_{\max} = 40.0$ |

**Shared Network Components:** Both architectures use Fourier-encoded positions $\hat{\xi} \in \mathbb{R}^N$ with frequencies $\omega_k = k\pi$ for $k \in \{1, 2, 4, 8\}$ as input to the trunk. Representations are fused via concatenation followed by a 32-dimensional dense layer to produce the control outputs.

### B.2 Training Hyperparameters

Models are trained using the Adam optimizer with an initial learning rate of $10^{-3}$ and exponential decay (decay rate 0.5 every 2000 steps). We utilize gradient clipping at a global norm of 1.0, a batch size of 32, and 500 epochs.

The composite loss function is defined as:

$$\mathcal{L} = \lambda_{\text{track}} l_{\text{track}} + \lambda_{\text{effort}} l_{\text{effort}} + \cdots + \lambda_{\text{accel}} l_{\text{accel}} \tag{29}$$

The coefficients and definitions are as follows:

- **Tracking ($\lambda_{\text{track}} = 5.0$):** Mean squared error between the PDE state $z$ and target $z_{target}$.
- **Effort ($\lambda_{\text{effort}} = 0.001$):** Regularization of control inputs defined as $\|u\|^2 + 0.1\|v\|^2$.
- **Boundary ($\lambda_{\text{bound}} = 100.0$):** Quadratic penalty for agents exiting the $[0.02, 0.98]$ domain.
- **Collision ($\lambda_{\text{coll}} = 1.0$):** Rejection penalty for agents within a safety radius $R_{safe} = 0.05$.
- **Acceleration Smoothing ($\lambda_{\text{accel}} = 0.1$):** Penalizes sudden changes in velocity, calculated as the squared difference $\|\Delta v\|^2$ between successive time steps.

### B.3 Numerical Discretization

The PDE dynamics are discretized using $N_x = 100$ grid points with a spatial step $\Delta x = 0.01$ (over a unit domain) and a temporal step $\Delta t = 0.001$ for $T = 300$ steps.

- **Heat Equation:** Integrated via a standard Crank-Nicolson scheme.
- **FKPP Equation:** Integrated via operator splitting. An explicit step handles reaction and forcing: $z^* = z^n + \Delta t[\rho z^n(1 - z^n) + B^n]$. This is followed by an implicit diffusion step solved via a tridiagonal matrix: $(I - rL)z^{n+1} = z^*$, where $r = \nu \Delta t / \Delta x^2$. Finally, states are clipped to the biologically relevant range $[0, 1]$.

## C Evaluation Setup

All numerical experiments and model training were conducted on a high-performance mobile workstation (Lenovo Legion). The specific hardware and software configurations are detailed below.

### C.1 Hardware Configuration

Experiments were executed on an **Intel(R) Core(TM) Ultra 9 275HX** processor featuring 24 physical cores and a maximum clock speed of 5.4 GHz. High-speed parallel computations for the PDE solvers and neural networks were offloaded to an **NVIDIA GeForce RTX 5090 Laptop GPU** with 24 GB of dedicated GDDR7 VRAM and a peak SM clock rate of 3090 MHz.

### C.2 Software Environment

The computational environment was hosted on **Windows Subsystem for Linux (WSL2)** running Ubuntu 22.04 with Linux Kernel 6.6.87. We utilized the **JAX** framework for high-performance numerical computing and hardware acceleration. The software stack versioning is summarized in Table 5.

**Table 5:** Software stack and library versions used for evaluation.

| **Component** | **Version / Specification** |
|---|---|
| OS | Ubuntu 22.04.x via WSL2 |
| NVIDIA Driver | 581.57 |
| JAX Backend | CUDA (GPU) |
| JAX Version | 0.8.1 |
| Tesseract-Core | 1.2.0 |
| Tesseract-JAX | 0.2.2 |
| Optimization | Optax (latest) |