EXTENDING FOURIER NEURAL OPERATORS FOR PARAMETERIZED AND COUPLED PDES

Anonymous authors
Paper under double-blind review

ABSTRACT

Parameterized and coupled partial differential equations (PDEs) are central to modeling phenomena in science and engineering, yet neural operator methods that address both aspects remain limited. We extend Fourier neural operators (FNOs) with minimal architectural modifications along two directions. For parameterized dynamics, we propose a hypernetwork-based modulation that conditions the operator on physical parameters. For coupled systems, we conduct a systematic exploration of architectural choices, examining how operator components can be adapted to balance shared structure with cross-variable interactions while retaining the efficiency of standard FNOs. Evaluations on benchmark PDEs, including the one-dimensional capacitively coupled plasma equations and the Gray–Scott system, show that our methods achieve up to 55$\sim$72% lower errors than strong baselines, demonstrating the effectiveness of principled modulation and systematic design exploration.

1 INTRODUCTION

Numerical simulation has long served as a cornerstone of scientific and engineering inquiry, underpinning advances in areas ranging from fluid dynamics (Ferziger et al., 2019) and climate modeling (Bauer et al., 2015) to material science (Rappaz et al., 2003) and structural analysis (Macneal & Harder, 1985). High-fidelity simulations, while indispensable, often entail substantial computational cost. These costs become especially significant when conducting parameter studies, uncertainty quantification, and real time decision making. As a result, surrogate modeling has emerged as an essential tool to expedite simulation workflows, enabling rapid approximations that preserve first-order fidelity while dramatically reducing computational burden. Recently, deep learning techniques have gained increasing traction in this domain. Methodologies such as physics-informed neural networks (PINNs) (Raissi et al., 2019; Karniadakis et al., 2021) and neural operators (NOs) (Li et al., 2020a;b; Lu et al., 2021) offer flexible, data-driven approximations of parametrized partial differential equations (PDEs), often delivering orders-of-magnitude speedups over traditional solvers while maintaining acceptable accuracy.

Despite significant progress in NO, most existing studies have considered scenarios with both varying initial conditions and parameter regimes, yet they often overlook the particular, yet prevalent case of parameterized dynamics with fixed initial conditions, typified by equations of the form $u_t = f(u; \mu)$ with $u(0) = u_0$. Such settings are ubiquitous in engineering, where model dynamics depend on physical parameters (e.g., forcing terms in inviscid Burgers' equation (Rewieński, 2003; Carlberg et al., 2013), diffusivity/reaction constants in chemically reacting flows (Buffoni & Willcox, 2010; Lee & Carlberg, 2020), and Reynolds number and viscosity in fluid dynamics (Stabile & Rozza, 2018)) while initial states remain unchanged across simulations. Studying this regime is important for practical applications, for example, to understand how variations in material properties, operating conditions, or control parameters affect system behavior from a fixed, well-defined state.

In this work, we focus specifically on NOs for parameterized coupled PDEs in fixed-initial-condition settings. To facilitate this exploration, we build upon Fourier neural operators (FNOs) (Li et al., 2021), a method now well-established in the community for its expressive parametric kernel in Fourier space. Our approach is designed to remain general and modular, making it adaptable to both FNO variants and other neural operator frameworks. To incorporate parameterized dynamics effectively, we introduce an architectural modification in which the meta-physics-knowledge

encoded in the governing equations is captured by a larger set of shared base parameters, while parameter-specific variations are represented through a smaller set of task-dependent model parameters (with a task being associated a specific physical parameter). To ensure the overall model remains lightweight, we implement this design using a compact hypernetwork, enabling parameterization while retaining a model size comparable to standard, non-parameterized FNOs.

Further, we extend the (parameterized) FNOs to effectively model the coupled systems while being efficient in the model size. While existing efforts (Xiao et al., 2023) introduce specialized decompositions stream to capture cross-field interactions, there is little systematic guidance of the design principles that govern effective cross-variable coupling in neural operators, particularly when aiming to preserve the efficiency and scalability of standard FNOs. In this paper we address that gap: we present a principled design space for extending FNOs to coupled problems, investigate the lift, Fourier layers, and projection operators for identifying effective locations and mechanisms for enabling cross-variable coupling.

In addition, we introduce a new benchmark PDE based on a simplified one-dimensional capacitively coupled plasma (CCP) model. This system is of practical importance in plasma physics and semiconductor manufacturing, and it exhibits rich parameterized dynamics that challenge existing operator-learning methods. Our formulation provides a tractable yet representative setting for evaluating neural operators under coupled and parameterized conditions.

Our contributions are summarized as

- Parametric extensions of FNOs for effectively modeling parameterized dynamics,
- Systematic dissection and adaptation of FNO architecture components for coupled systems,
- Introduction of a novel benchmark problem with many physical parameters describing simplified sheath dynamics governed by plasma physics, and
- Extensive experimentation with the proposed methods, comparisons with baselines, and validation on benchmark problems.

2 Technical Background

2.1 Neural Operators

NOs are a type of data-driven surrogate model designed to learn mappings between functions rather than traditional input-output pairs, making them particularly effective for problems governed by PDEs (Lu et al., 2021; Kovachki et al., 2023; Boullé & Townsend, 2024; Azizzadenesheli et al., 2024). By taking discrete representations of continuous functions as input, NOs produce functional representations that can efficiently approximate and simulate complex physical systems.

Let $\mathcal{A}$ and $\mathcal{U}$ be two function spaces, and consider a nonlinear operator $G : \mathcal{A} \to \mathcal{U}$ mapping between them. Neural operators are a class of machine learning methods designed to construct a surrogate $G_\Theta \approx G$ within a trial space of neural networks. Given input-output pairs $\{(a^i, u^i)\}_{i=1}^n$, where $x^i \in \mathcal{A}$ and $u^i = G(a^i) \in \mathcal{U}$, a neural operator can be trained in the standard supervised manner by solving: $\min_\Theta \sum_{i=1}^n L\left(G_\Theta(a^i), u^i\right)$, where L denotes a loss term and Θ denotes the parameters of the neural operator G_Θ.

In practice, the functional input data a^i are first discretized; for example, if $a(x)$ and $u(x)$ are functions, the grid representation $\boldsymbol{a} = [a(x_1), \ldots, a(x_m)]^\mathsf{T} \in \mathbb{R}^{m \times d_a}$ serves as input along with an evaluation point (or set of points) $D = \{x_1, \ldots, x_m\}$ in the domain of a, and the neural operator learns a mapping $\boldsymbol{a} \mapsto G_\Theta(\boldsymbol{a})$ that predicts $G_\Theta(\boldsymbol{a})(x_p) \approx u(x_p)$ (Lu et al., 2021; Li et al., 2021). With the assumption that the spatial grid where $a(x)$ and $u(x)$ are discretized is fixed, $(D, \boldsymbol{a}) \mapsto \boldsymbol{u}$ (under G) produces a vector $\boldsymbol{u}$ of operator evaluations, and therefore samples $(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{u})$ form the training data for the NO learning problem.

2.2 Fourier neural operators

FNOs are a specific instantiation of NOs that model learnable nonlinear kernel integral operators, with the integrals efficiently evaluated in the Fourier frequency domain. FNOs, $G_\theta : \mathcal{A} \to \mathcal{U}$, first projects the input data into hidden representations via local transformation, $v_0(x) = P(a(x))$, where P is typically chosen as a linear projector in practice. FNOs then employ the iterative updates of

hidden representations $v_\ell \mapsto v_{\ell+1}$ via

$$v_{\ell+1}(x) \coloneqq \sigma\Big(W v_\ell(x) + \left(\mathcal{K}(a;\phi)v_\ell\right)(x)\Big), \qquad \forall x \in D$$

where W is a linear transformation and $\mathcal{K}$ denotes a kernel integral operator. The kernel integral operator is defined as

$$\left(\mathcal{K}(a;\phi)v_\ell\right)(x) \coloneqq \int_D \kappa(x, y, a(x), a(y); \phi) v_\ell(y) \mathrm{d}y, \qquad \forall x \in D \tag{1}$$

where ϕ indicates the learable parameters that characterize the kernel. For efficient computation of Eq. (1), FNOs define the kernel integral operator in Fourier space and perform a convolution operation in that space such that

$$\left(\mathcal{K}(\phi)v_\ell\right)(x) = \mathcal{F}^{-1}\Big(R_\phi \cdot (\mathcal{F}v_\ell)\Big)(x), \qquad \forall x \in D,$$

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote Fourier and inverse Fourier transformations, respectively, and R_ϕ denotes the Fourier transform of function κ. In practice, R_ϕ is parameterized as a learnable tensor. The last hidden representation $v_T(x)$ is then projected back to the target data space via a local transformation, $u(x) = Q(v_T(x))$, where Q is typically modeled as a shallow multi-layer perceptron (MLP).

3 METHODS

Our methods seek minimal yet principled architectural modifications to FNOs that enable them to handle parameterized and coupled dynamical systems. We emphasize minimal modification as a design principle: this ensures (i) the resulting models preserve the efficiency and scalability of FNOs and (ii) performance gains can be attributed to structural insights rather than increased capacity.

To this end, we focus on the three core computational components of FNOs: the lift operator (input projection), the Fourier layers (spectral updates), and the projection operator (output mapping). We systematically explore how to adapt each component to encode parameter dependence and to capture cross-variable interactions in coupled systems. The remainder of this section presents these extensions in two parts: one subsection introduces the parameterized extensions based on hypernetwork-driven modulation, and another subsection describes the design spaces for coupled extensions that capture interactions among multiple PDE components.

3.1 EXTENSIONS TO PARAMETRIZED FNO

A straightforward way to extend FNOs for handling physical parameters, $\mu \in \mathbb{R}^{n_\mu}$, is input augmentation. In this approach, the parameters are concatenated with the function input as $[a(x); \mu(x)]$, where $\mu(x) = \mu$ is constant across the spatial domain. The lift operator then encodes this augmented input into a latent representation, $v_0(x) = P([a(x), \mu(x)])$. This strategy mirrors prior work, for example in the neural ODE community, where augmenting the input has been shown to enhance the expressivity of the learned dynamics (Dupont et al., 2019; Massaroli et al., 2020; Lee & Parish, 2021), and physics-informed neural networks (Cai et al., 2021; Cho et al., 2024). We refer to this variant using the prefix "p" (e.g., `pFNO`) to denote parameterization via input augmentation.

While simple and effective in some cases, input augmentation may not fully capture how physical parameters influence the underlying dynamics. To address this, we draw inspiration from recent advances in implicit neural representations (INRs) (Sitzmann et al., 2020; Fathony et al., 2020; Dupont et al., 2022) and PINNs (Cho et al., 2023). These works demonstrate that modulation, i.e., adjusting network parameters conditioned on side information, provides a powerful mechanism to encode complex dependencies without substantially increasing model size.

Following this principle, we design a hypernetwork-based modulation scheme. A compact hypernetwork (Ha et al., 2017) takes the physical parameters μ as input and outputs a set of shifts, $s(x, \mu) = f^{\text{hyper}}(x, \mu)$, where $s(x, \mu) = [s_1(x, \mu), \ldots, s_L(x, \mu)]$ corresponds to layer-specific adjustments. These shifts are incorporated as additional biases in the Fourier layers:

$$v_{\ell+1}(x) \coloneqq \sigma\Big(W v_\ell(x) + \left(\mathcal{K}(a;\phi)v_\ell\right)(x) + s_\ell(x, \mu)\Big). \qquad \forall x \in D$$

We refer to this variant using "hp" (e.g., `hpFNO`) indicating the hypernetwork-based approach.

The rationale behind this design is twofold. First, modulation has already proven effective in INRs and PINNs, where conditioning on auxiliary variables enables networks to represent rich families of functions without retraining. Second, unlike input augmentation, modulation operates directly on the model's internal representations, allowing parameter dependence to influence the dynamics at every stage of computation rather than only at the input. Together, these factors suggest that the hp-variant should more effectively capture parameterized dynamics while remaining lightweight.

3.2 Extensions to coupled FNO

We now turn to extending FNOs for coupled systems, where multiple interdependent variables evolve under shared dynamics. For the simplicity of exposition, we focus on the case of two variables and differentiate each component by superscript, $z^{\square}$ with a generic variable z, where $\square \in \{\alpha, \beta\}$. Our design philosophy follows the same principle outlined at the beginning of this section: to retain the minimal and scalable structure of standard FNOs while introducing only the modifications necessary to capture cross-variable interactions. To this end, we investigate two main questions: (1) within the core computational components of FNOs (the lift operator, P, Fourier layers, and projection operator, Q), should the transformations be shared across variables or separate for each? and (2) at what stages, and in what form, should information mixing occur between variables to most effectively model their coupling?

Before expanding on the design exploration, we first define the setup. Our neural operator is trained to learn a mapping $(u^{\alpha}(x), u^{\beta}(x) = G_{\Theta}(a^{\alpha}(x), a^{\beta}(x))$, where (a^{α}, a^{β}) denote the inputs associated with the two variables and (u^{α}, u^{β}) denote their corresponding outputs.

Lift operator, P The local transform, $(Pa)(x) = v_0(x) \in \mathbb{R}^{d_v}$ takes a discrete representation of the input function. In the coupled setting, we consider two main design choices that differ in whether the operator is shared or individual across the two variables.

(P1) The lift operator $P : \mathbb{R}^1 \to \mathbb{R}^{d_v}$ is shared across the two input components, so that both variables are mapped into the latent space using the same transformation.
(P2) Alternatively, the lift operator consists of two separate mappings, $P = (P^{\alpha}, P^{\beta})$, each specific to one input component, allowing the two variables to have independent latent representation.

Fourier layers The Fourier layer largely consists of two components where we can make design choices: the point-wise linear map $Wv_{\ell}(x)$ and global spectral convolution $\mathcal{K}v_{\ell}(x)$.

(L1) and (L2): The point-wise linear map can be implemented either as a single operator shared by both variables ((L1)) or as two separate operators (W^{α} and W^{β}) ((L2)).

(G) For coupled systems, we define the global spectral convolution to perform coupling only in Fourier space. Specifically, each variable is first transformed independently, $\tilde{v}^{\square}(k) = \mathcal{F}v_{\ell}^{\square}(k)$, for $\square = \{\alpha, \beta\}$. Their spectral representations are then combined through a shallow encoder network $f^{\text{enc}}(\tilde{v}^{\alpha}(k), \tilde{v}^{\beta}(k))$, followed by the usual mode-wise kernel multiplication $\tilde{\tilde{v}}(k) = R_{\phi}(k)\tilde{v}(k)$. The result is decomposed into variable-specific coefficients using another shallow network, $(\tilde{\tilde{v}}^{\alpha}(k), \tilde{\tilde{v}}^{\beta}(k) = f^{\text{dec}}(\tilde{\tilde{v}}(k)$, and finally mapped back to the data space via the inverse Fourier transform, $v_{\ell+1}^{\square}(k) = \mathcal{F}^{-1}(\tilde{\tilde{v}}^{\square}(k))$, for $\square = \alpha, \beta$.

A key design choice here is that coupling is introduced only in Fourier space, after the individual Fourier transforms. This has several benefits compared to approaches that use entirely separate convolution operators and exchange hidden representations across variables. First, it preserves the core structure of FNOs, keeping the model lightweight and scalable. Second, the Fourier domain naturally captures long-range correlations, making it a well-suited location for cross-variable mixing. Finally, by avoiding repeated exchanges of hidden states in the spatial domain, this design reduces computational overhead while maintaining symmetry between the two variables.

Projection operator, Q The local transform, $(Qv_T)(x) = u(x)$, resembles with that of the lift operator P and, thus, similar choices apply. We focus here on the single-channel output format for each variable, which is the natural setting for coupled systems.

(Q1) The projection operator is shared across both output components, $Q : \mathbb{R}^{d_v} \to \mathbb{R}^1$.
(Q2) The projection operator is defined separately for each output component, $Q = (Q^\alpha, Q^\beta)$, with $Q^\square : \mathbb{R}^{d_v} \to \mathbb{R}^1$ for $\square \in \alpha, \beta$.

To refine the design space under (Q2), we adopt the adaptive basis viewpoint (Cyr et al., 2020), which interprets the last hidden activation as a set of adaptive basis functions, $\Psi(a(x)) = [\psi_1(a(x)), \ldots, \psi_{n_q}(a(x))]^\mathsf{T}$, and the weight of the output layer as coefficients, $\Xi = [\xi_1, \ldots, \xi_{n_q}]^\mathsf{T}$: $u(x) = \sum_{i=1}^{n_q} \xi_i \psi_i(x)$. In practice, Q is parameterzed as a shallow MLP, $(Qv_T)(x) = W_2\sigma(W_1 v_T(x) + b_1) + b_2$ and the adaptive basis corresponds to $\Psi(a(x)) = \sigma(W_1 v_T(x) + b_1)$ and the coefficients to $\Xi = W_2$. The shared/separate choices are then realized by sharing or splitting the weight matrices and biases.

From this perspective, Q can differ in whether the basis and the coefficients are shared or separate: (Q2a) Shared basis and separate coefficients, (Q2b) Separate basis and shared coefficients, and (Q2c) Separate basis and separate coefficients.

3.3 Putting all together

Based on our experimentation (presented in Appendix D.1.1) with the different combinations of the design choices (further elaborated in Appendix A), we define our model, extended FNOs — $\texttt{FNO}_\times$, as a realization of the combination: (P1) + (L2) + (G)+ (Q2c), which employs the shared lift operator, the separate point-wise linear maps, the proposed coupled global spectral convolution layer, and the separate sets of basis functions and separate sets of coefficients in the projection operator.

4 Experimental results

In this section, we present experimental results comparing the proposed methods with a range of baseline models. All methods are implemented in Python with PyTorch (Paszke et al., 2019), building upon the original implementation of FNOs (Li et al., 2021) and extending it to handle coupled and parameterized settings. Each experiment is repeated five times with different random seeds, and all computations are performed on an Nvidia A100 GPU with 80GB memory.

4.1 Setup

Tasks and training We consider time-dependent PDEs that generate solution trajectories $\{u(x, t; \mu)\}_{t=0}^{T}$ with $u(x, 0; \mu) = u(0)$. NOs are formulated as one-step evolution maps that approximate the discrete-time flow of the underlying PDE. Specifically, given a finite history of past solution states $\{u(t-\tau)\}_{\tau=0}^{T_{\text{in}}-1} \in \mathcal{A}$, the operator predicts the next state $u(t+1) \in \mathcal{U}$: $G_\Theta : \mathcal{A} \to \mathcal{U}$. Multi-step forecasts are obtained autoregressively by recursively applying the same operator. During training and testing, the initial input window $(u(T_{\text{in}} - 1), \ldots, u(0))$ is assumed to be available, while subsequent rollouts incorporate predicted states back into the input window.

Baselines As baselines of comparisons, we consider methods inherited from Xiao et al. (2023):

- $\texttt{FNO}_\texttt{c}$: FNOs taking inputs as a vertical concatenation of the discretized input data,
- CFNO: Two separate FNOs sharing exchanging hidden representation in Fourier layers,
- $\texttt{MWT}_\texttt{c}$: MWTs (Gupta et al., 2021) taking a vertically concatenated input data,
- CMWNO: Multiwavelet NOs specifically designed for coupled systems (Xiao et al., 2023),

and other relevant baselines, DeepONets (Lu et al., 2021) and U-Nets (Ronneberger et al., 2015), modified to handle multi input:

- $\texttt{DON}_\texttt{c}$: DeepONets extended to take a vertical concatenation of the discretized input data,
- $\texttt{U-Net}_\texttt{c}$: U-Nets taking two-channel input.

and, finally, one variant of a certain combinations of choices from the described design choices:

- $\texttt{FNO}_\mathbb{m}$: the lift operator projects the two-channel input (a^α, a^β) jointly into a shared latent space, i.e., $P : \mathbb{R}^2 \to \mathbb{R}^{d_v}$ with $v_0(x) = P([a^\alpha(x), a^\beta(x)])$.

We refer to Appendix B.1 for the detailed description of the baselines.

Performance evaluation We evaluate performance using the normalized root mean square error (nRMSE), averaged over all test trajectories:

$$\text{nRMSE} = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \left\| u_{0:T}^{(i)} - \tilde{u}_{0:T}^{(i)} \right\|_2 \Big/ \left\| u_{0:T}^{(i)} \right\|_2$$

where u and $\tilde{u}$ denote the ground-truth and predicted trajectories. For coupled systems, nRMSE is computed separately for each variable, and the final error is the sum: $\text{err} = \text{nRMSE}^{\alpha} + \text{nRMSE}^{\beta}$.

4.2 1-DIMENSIONAL CAPACITIVELY COUPLED PLASMA FLUID MODEL

In this section, we look at a 1-dimensional capacitively coupled plasma (CCP) model as an example to demonstrate the performance of the proposed model in a parameteric setting.

The simplified CCP model describes a low-temperature plasma that is generated between two electrodes when an alternating voltage is applied, operating in a manner similar to a capacitor. In this system, the oscillating electric field accelerates electrons, which then collide with neutral gas atoms or molecules, producing ions and additional electrons that sustain the plasma. CCPs are widely used in microelectronics manufacturing, including etching and thin-film deposition for semiconductor devices, as well as in materials processing and surface engineering (Lieberman & Lichtenberg, 1994; Rauf et al., 2023).

The governing equation is defined as follows:

$$\partial_t n_{\text{e}} = -\partial_x \Gamma_{\text{e}} + R, \qquad \text{(Electron continuity equation)}$$
$$\partial_{xx}\phi = -e/\epsilon_0 (n_{\text{e}} - n_{\text{io}}), \quad \text{(Poisson equation)}$$

where $n_{\text{e}}(x,t)$ and $\phi(x,t)$ denote electron density and electric potential, which are the solutions of the equations, and $\partial_{\square}$ refers to a partial derivative with respect to $\square$. Γ_{e} is electron flux, defined as diffusion flux and drift flux such as $\Gamma_e = -D\partial_x n_{\text{e}} - \mu n_{\text{e}} \partial_x \phi$ with electron diffusion coefficient D and electron mobility coefficient μ, and ion density is defined as $n_{\text{io}} = R_0(x_2 - x_1)\sqrt{m_i/eT_{\text{e}}}$. R and R_0 denote reaction rate and coefficient, respectively. The ion density is assumed constant in this model.

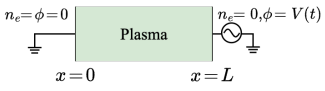


Figure 1: The spatial domain and boundary conditions.

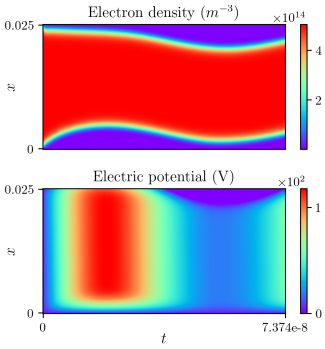


Figure 2: An example solution snapshot of electron density and electric potential.

Physical parameters There are several physical parameters characterizing a domain geometry, boundary conditions (BCs), and governing physical dynamics (See Table 4 in Appendix). Among them, in this study, we consider fixed parameters for geometry (illustrated in Figure 1), but varying parameters that define the boundary conditions and the dynamics. In the BCs, the zero boundary condition is given to the electron density (at $x = 0, L$). The electric potential satisfies a Dirichlet condition (at $x = 0$), and a time-periodic Dirichlet condition at $x = L$, $V(t) = V_0 \sin(2\pi t)$. That is, the potential at the right boundary oscillates sinusoidally with amplitude V_0 (driving voltage) and frequency f and we vary V_0 in the experiments. This models the externally applied radio-frequency forcing that sustains the discharge in CCP. Along with V_0, we consider two other physical parameters that governs the dynamics: reaction coefficient, R_0, and ion mass, m_i. Figure 2 depicts solution snapshots of $n_{\text{e}}(x,t)$ and $\phi(x,t)$, simulated over a single cycle, configured with $V_0 = 100$, $R_0 = 2.7 \times 10^{20}$, and $m_i = 40 \times 1.67 \times 10^{-27}$.

4.2.1 DATA SETUP

Data generation We collect solution snapshots by numerically simulating the governing equations using a finite-difference solver. The spatial domain is discretized into 128 cells, and the time domain into 100,000 steps per cycle, with the cycle determined by the boundary driving frequency of 13.56 MHz. A fixed initial condition is used to focus on scenarios where capturing physical dynamics is critical. We then subsample every 1000th snapshot, yielding 100 temporal indices per solution. Physical parameters are sampled on an equidistant grid, which contains 100 elements: $V_0 \in [100, 300]$, $R_0 \in [2.7 \times 10^{19}, 2.7 \times 10^{20}]$, and $m_i \in [1.67 \times 10^{-26}, 6.68 \times 10^{-26}]$.

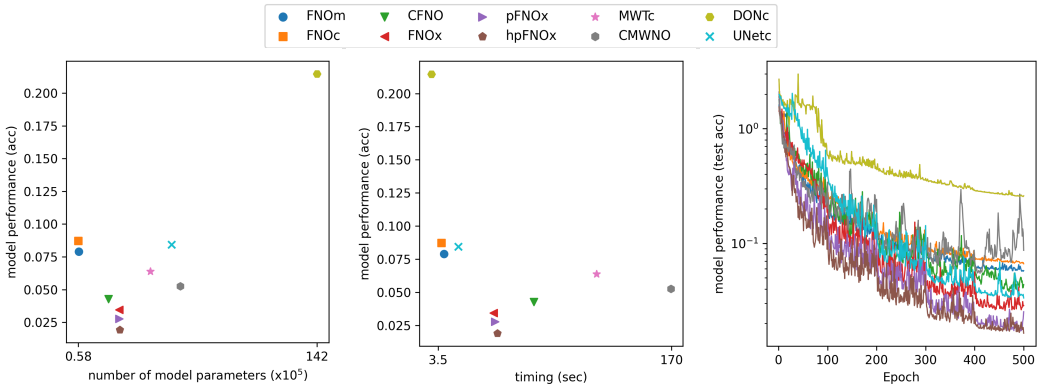
4.2.2 NUMERICAL RESULTS

1-dimensional parameter space We perform three separate experiments, each on a dataset obtained by varying a single physical parameter: R_0, V_0, or m_i. Table 1 summarizes the results, reporting prediction errors on the test set for the proposed methods and the baseline models. The proposed $\texttt{FNO}_\texttt{x}$ architecture ($\texttt{FNO}_\texttt{x}$, without the parametric extension) provides notable improvement over the baseline methods. The overall second best model in this set of experiments is CMWNO and, compared to its performance, $\texttt{FNO}_\texttt{x}$ achieve the prediction accuracy improved up to 38% (in the varying reaction rate scenario). The parametric extensions, $\texttt{pFNO}_\texttt{x}$ and $\texttt{hpFNO}_\texttt{x}$, further improve the performance, achieving up to 55% improvement over the best performing baselines.

Table 1: Performance comparisons between the proposed methods and the baseline methods. All models are tested on three parameters (R_0, V_0 and m_i) individually. The performance is measured in the relative ℓ^2-error (nRMSE); the reported numerical values refer to the mean ($\pm$ std. dev.).

Model	Reaction rate	Driving voltage	Ion mass
$\texttt{FNO}_\texttt{m}$	0.0403 ($\pm$ 0.0012)	0.0791 ($\pm$ 0.0243)	0.0363 ($\pm$ 0.0067)
$\texttt{FNO}_\texttt{c}$	0.0375 ($\pm$ 0.0055)	0.0873 ($\pm$ 0.0200)	0.0299 ($\pm$ 0.0030)
CFNO	0.0315 ($\pm$ 0.0093)	0.0428 ($\pm$ 0.0064)	0.0333 ($\pm$ 0.0076)
$\texttt{MWT}_\texttt{c}$	0.0409 ($\pm$ 0.0032)	0.0639 ($\pm$ 0.0038)	0.0403 ($\pm$ 0.0040)
CWMNO	0.0312 ($\pm$ 0.0023)	0.0526 ($\pm$ 0.0521)	0.0241 ($\pm$ 0.0036)
$\texttt{DON}_\texttt{c}$	0.0844 ($\pm$ 0.0058)	0.2147 ($\pm$ 0.0414)	0.1035 ($\pm$ 0.0174)
$\texttt{U-NET}_\texttt{c}$	0.1084 ($\pm$ 0.0345)	0.0844 ($\pm$ 0.0354)	0.1719 ($\pm$ 0.1022)
$\texttt{FNO}_\texttt{x}$	0.0193 ($\pm$ 0.0059)	0.0345 ($\pm$ 0.0108)	0.0212 ($\pm$ 0.0062)
$\texttt{pFNO}_\texttt{x}$	0.0194 ($\pm$ 0.0075)	0.0278 ($\pm$ 0.0053)	0.0142 ($\pm$ 0.0021)
$\texttt{hpFNO}_\texttt{x}$	**0.0154** ($\pm$ 0.0029)	**0.0192** ($\pm$ 0.0040)	**0.0128** ($\pm$ 0.0017)

Computational/memory aspects Since prediction accuracy alone does not provide a complete picture of the experimental results, Figure 3 presents additional information under the varying driving voltage scenario, including model size (3a) and computational timing (3b), alongside the models' prediction accuracy. The model size is measured in the number of trainable model parameters and the timing reports a per-epoch training time measured and averaged across the total number of epochs. The both figures provide the same observation: the proposed methods $\texttt{FNO}_\texttt{x}$, $\texttt{pFNO}_\texttt{x}$, and $\texttt{hpFNO}_\texttt{x}$ achieve improvement in accuracy without sacrificing the model size (i.e., memory footprint) and the computational wall time too much. Additionally, Figure 3c shows the test loss trajectories over the training epoch. Along with the computational timing per epoch (Figure 3b), the loss trajectories provide a complete picture on models' computational requirements in achieving the presented accuracy.



(a) Model size (b) Timing (c) Test loss

Figure 3: [1D CCP] Plots depict the number of model parameters versus the model performance (left), computational timing versus the model performance (middel), and the test loss trajectories over epochs (right). These results are from the varying driving voltage scenario.

Study on varying T_{in} Although the true dynamics depend on physical parameters, all non-parametric NOs in Table 1 achieve reasonable accuracy. This is because the NOs take as input a window of past snapshots, characterized by T_{in}, akin to delay embedding in dynamical systems (Takens, 2006). By consuming multiple consecutive time steps, the NOs implicitly encode information about system parameters through temporal correlations, enabling them to infer dynamics.

Thus, the benefits of using the parameterized version of $\texttt{FNO}_{\text{x}}$can be more pronounced when T_{in} becomes smaller. To see this, we extend the experimentation for varying $T_{\text{in}} = \{10, 5, 2, 1\}$. Table 2 reports the results comparing the performance of $\texttt{FNO}_{\text{c}}$ and $\texttt{FNO}_{\text{x}}$ and their two parametric variants (pFNOc, hpFNOc), and ($\texttt{pFNO}_{\text{x}}$, $\texttt{hpFNO}_{\text{x}}$), respectively. As T_{in} decreases, the overall model performance, which is expected. For $T_{\text{in}} = \{10, 5\}$, the improvement achieved by $\texttt{hpFNO}_{\text{x}}$ over $\texttt{FNO}_{\text{x}}$ over 20%. For $T_{\text{in}} = \{2, 1\}$, while the hp variants maintain the accuracy level around 13∼16% relative errors. Considering that only initial conditions are available in most practical setups, maintaining this level of performance could be considered as important step forward for FNOs.

Table 2: Performance comparisons of non-parameterized (FNOc, $\texttt{FNO}_{\text{x}}$) and parameterized (pFNOc, hpFNOc, $\texttt{pFNO}_{\text{x}}$, $\texttt{hpFNO}_{\text{x}}$) models for varying $T_{\text{in}} = \{10, 5, 2, 1\}$. All models are tested on the reaction rate case. The performance is measured in the relative ℓ^2-error; the reported numerical values refer to the mean (± std. dev.). † indicates that the models fail to learn the dynamics and ‡ indicates that the model performance are averaged from the best 5 runs out of 10 total runs.

Model	$T_{\text{in}} = 10$	$T_{\text{in}} = 5$	$T_{\text{in}} = 2$	$T_{\text{in}} = 1$
$\texttt{FNO}_{\text{c}}$	0.0375 (±0.0055)	0.1324 (±0.0304)	1.0048† (±0.6356)	1.4484† (±0.3128)
$\texttt{pFNO}_{\text{c}}$	0.0334 (±0.0101)	0.0923 (±0.0276)	0.2151‡ (±0.0374)	0.3757‡ (±0.0679)
$\texttt{hpFNO}_{\text{c}}$	0.0196 (±0.0021)	0.0804 (±0.0237)	0.1515‡ (±0.0070)	0.1609‡ (±0.0043)
$\texttt{FNO}_{\text{x}}$	0.0193 (±0.0059)	0.0406 (±0.0093)	1.2832† (±0.4298)	1.8143† (±0.0558)
$\texttt{pFNO}_{\text{x}}$	0.0194 (±0.0075)	0.0464 (±0.0158)	0.1640‡ (±0.0155)	0.2522‡ (±0.1376)
$\texttt{hpFNO}_{\text{x}}$	**0.0154** (±0.0029)	**0.0317** (±0.0022)	**0.1324**‡ (±0.0242)	**0.1372**‡ (±0.0100)

Adaptive basis view point In the next set of experiments, we investigate further on the adaptive basis viewpoint given on the last projection operator Q. We first vary the width (that is, the number of basis functions) of the last layer of Q for $n_q = \{8, 16, 32, 64, 128\}$ and measure the prediction performance of $\texttt{hpFNO}_{\text{x}}$ on the varying reaction rate scenario. Taking the $d_{\text{basis}} = 8$ as a baseline, Figure 4 reports the percentage of improvements (shown in the bottom filled markers). Next, we test the idea of having (close to) orthogonal basis functions and repeat the same experiments. An orthonormality condition imposed to the learned basis, $\Psi(a(x))$; we could achieve further improvement (shown in the top filled markers). As d_{basis} becomes small, the effect of the orthogonality enforcement becomes higher, leading to ∼10% improvements. We define the orthogonality of the learned basis functions and its computation in Appendix B.3.

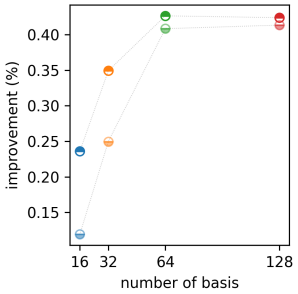


Figure 4: Num. of bases v.s. performance improvement.

2-dimensional parameter space Next we extend our investigation to 2D input parameter space, a parameter space spanned by the reaction rate and the driving voltage. For input parameter sampling, we consider the same ranges for each parameter $V_0 \in [100, 300]$, $R_0 \in [2.7 \times 10^{19}, 2.7 \times 10^{20}]$, construct a uniform 2D mesh grid (21×21), and randomly split them into training and test sets with 9:1 ratio. We can make observations that are similar with the ones from the 1D input parameter cases; the proposed architecture $\texttt{FNO}_{\text{x}}$ provides improvement in prediction accuracy and parameterized versions of $\texttt{FNO}_{\text{c}}$ and $\texttt{FNO}_{\text{x}}$ further reduce the prediction errors, leading to the best performance with $\texttt{hpFNO}_{\text{x}}$. The model size and the computational timing follow the similar trends with the 1D

Table 3: Performance comparisons for the 2D input parameter case.

Model	Performance
$\texttt{FNO}_{\text{m}}$	0.0285 (± 0.0028)
$\texttt{FNO}_{\text{c}}$	0.0252 (± 0.0026)
CFNO	0.0223 (± 0.0023)
$\texttt{MWT}_{\text{c}}$	0.0305 (± 0.0028)
CMWNO	0.0298 (± 0.0023)
$\texttt{DON}_{\text{c}}$	0.0409 (± 0.0077)
$\texttt{U-Net}_{\text{c}}$	0.0461 (± 0.0160)
$\texttt{FNO}_{\text{x}}$	0.0163 (± 0.0009)
$\texttt{pFNO}_{\text{x}}$	0.0140 (± 0.0012)
$\texttt{hpFNO}_{\text{x}}$	**0.0124** (± 0.0011)

case, only except the slightly increased per-epoch computation time due to the increased number of data instances in the dataset for all methods.

4.3 Gray–Scott equations

As the second benchmark problem, we consider the Gray–Scott equations, which form a system of coupled reaction-diffusion equations describing the spatio-temporal dynamics of interacting chemical species. We consider a parameterized dynamics of this equation, where the parameters are diffusion coefficients and feed rate. The task of the NOs are performing predictions of time-dependent PDE solutions in the auto-regressive manner, and we follow the same training/testing protocols shown in the first benchmark problem. We refer to Appendix E for details.

We observe that the proposed methods achieve 54% improvement (for the varying diffusion coefficients scenario) and 72% improvement (for the varying feed rates scenario) in terms of the prediction accuracy compared to the performance of the best performing baseline method. We refer to Appendix E.1 for the details of the equations, the experimental setups and results.

5 Related work

Building on the success of FNOs, there have been many variants extending their capabilities. Physics-informed neural operators (PINO) hybridize data supervision with PDE residuals to enforce physical constraints (Li et al., 2024). Geo-FNO learns a deformation from an arbitrary physical geometry into a latent regular grid so that the FFT-based FNO can be applied, making the spectral approach applicable to irregular domains and nonuniform discretizations (Li et al., 2023).

More relevant to our proposed approach, several variants have focused on enhancing model performance through architectural improvements. Factorized FNO revisits spectral layers and introduces separable spectral operators and improved residual connections that enable deeper networks and better accuracy across simulation benchmarks (Tran et al., 2023). U-NO proposes a U-shaped, memory-enhanced neural operator architecture that borrows the encoder–decoder with skip connections to permit deeper operator nets with improved training stability (Rahman et al., 2023). Similarly, U-FNO combines Fourier layers with U-Net–style multiscale processing, enhancing expressivity while maintaining resolution independence (Wen et al., 2022). There have been non-Fourier-based NOs such as Multiwavelet NOs (Gupta et al., 2021) and Laplace NOs (Cao et al., 2024), for example.

Another line of relevant research in NOs is the study of coupled PDEs, which remains relatively underexplored. The coupled multiwavelet neural operator (CMWNO) is among the first to address this direction, introducing a framework that decouples integral kernels during multiwavelet decomposition and reconstruction to efficiently capture interactions in coupled PDE systems (Xiao et al., 2023). More recently, attention-based approaches have been proposed for multiphysics PDEs (Rahman et al., 2024), focusing on large-scale pretraining and transfer. In the parameterized setting, another work (Subramanian et al., 2023) has explored scaling laws and transfer behavior under very large datasets and compute budgets. Compared to these methods, our work emphasizes systematic architectural enhancements to FNOs that capture parameterized dynamics and coupled interactions with minimal modifications and without reliance on large-scale training budgets.

6 Conclusion

This work has investigated extensions of Fourier neural operators (FNOs) for modeling parameterized and coupled time-dependent partial different equations. To enhance the capability of FNOs in handling parameterized dynamics, we have proposed a novel modulation-based architectural modification that incorporates parameter information in a lightweight yet effective manner. For coupled systems, we have defined design spaces of FNOs and conducted an in-depth systematic investigation to identify economic architectures that preserve efficiency while capturing inter-component interactions. In addition, we have introduced a novel benchmark problem on plasma physics, which introduces rich parameterized dynamics and contributes a valuable testbed to the community. Experimental results have demonstrated that our proposed approaches achieve significant improvements in predictive performance, while maintaining model size and runtime efficiency comparable to standard FNOs.

7 ETHICS STATEMENT

This study focuses on methodological advances in machine learning for computational physics problems, and we do not anticipate any significant ethical concerns arising from the proposed approaches.

8 REPRODUCIBILITY STATEMENT

To support reproducibility, we provide detailed descriptions of the proposed methods, including model architectures, algorithms, loss definitions, hyperparameters, training setups, and baseline comparisons. All experiments are described in the main text and appendix, and upon acceptance we will release the full source code to further facilitate reproduction of our results.

REFERENCES

Kamyar Azizzadenesheli, Nikola Kovachki, Zongyi Li, Miguel Liu-Schiaffini, Jean Kossaifi, and Anima Anandkumar. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, 6(5):320–328, 2024.

Peter Bauer, Alan Thorpe, and Gilbert Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525(7567):47–55, 2015.

Nicolas Boullé and Alex Townsend. A mathematical guide to operator learning. In *Handbook of Numerical Analysis*, volume 25, pp. 83–125. Elsevier, 2024.

Marcelo Buffoni and Karen Willcox. Projection-based model reduction for reacting flows. In *40th Fluid Dynamics Conference and Exhibit*, pp. 5008, 2010.

Shengze Cai, Zhicheng Wang, Sifan Wang, Paris Perdikaris, and George Em Karniadakis. Physics-informed neural networks for heat transfer problems. *Journal of Heat Transfer*, 143(6):060801, 2021.

Qianying Cao, Somdatta Goswami, and George Em Karniadakis. Laplace neural operator for solving differential equations. *Nature Machine Intelligence*, 6(6):631–640, 2024.

Kevin Carlberg, Charbel Farhat, Julien Cortial, and David Amsallem. The GNAT method for nonlinear model reduction: Effective implementation and application to computational fluid dynamics and turbulent flows. *Journal of Computational Physics*, 242:623–647, 2013.

Woojin Cho, Kookjin Lee, Donsub Rim, and Noseong Park. Hypernetwork-based meta-learning for low-rank physics-informed neural networks. *Advances in Neural Information Processing Systems*, 36:11219–11231, 2023.

Woojin Cho, Minju Jo, Haksoo Lim, Kookjin Lee, Dongeun Lee, Sanghyun Hong, and Noseong Park. Parameterized physics-informed neural networks for parameterized PDEs. In *International Conference on Machine Learning*, pp. 8510–8533. PMLR, 2024.

Eric C Cyr, Mamikon A Gulian, Ravi G Patel, Mauro Perego, and Nathaniel A Trask. Robust training and initialization of deep neural networks: An adaptive basis viewpoint. In *Mathematical and Scientific Machine Learning*, pp. 512–536. PMLR, 2020.

Tobin A Driscoll, Nicholas Hale, and Lloyd N Trefethen. Chebfun guide, 2014.

Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented neural odes. *Advances in neural information processing systems*, 32, 2019.

Emilien Dupont, Hyunjik Kim, SM Ali Eslami, Danilo Jimenez Rezende, and Dan Rosenbaum. From data to functa: Your data point is a function and you can treat it like one. In *International Conference on Machine Learning*, pp. 5694–5725. PMLR, 2022.

Rizal Fathony, Anit Kumar Sahu, Devin Willmott, and J Zico Kolter. Multiplicative filter networks. In *International conference on learning representations*, 2020.

Joel H Ferziger, Milovan Perić, and Robert L Street. *Computational Methods for Fluid Dynamics*. springer, 2019.

Gaurav Gupta, Xiongye Xiao, and Paul Bogdan. Multiwavelet-based operator learning for differential equations. *Advances in neural information processing systems*, 34:24048–24062, 2021.

David Ha, Andrew M Dai, and Quoc V Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, 2015. URL https://arxiv.org/abs/1412.6980.

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

Kookjin Lee and Kevin T Carlberg. Model reduction of dynamical systems on nonlinear manifolds using deep convolutional autoencoders. *Journal of Computational Physics*, 404:108973, 2020.

Kookjin Lee and Eric J Parish. Parameterized neural ordinary differential equations: Applications to computational physics problems. *Proceedings of the Royal Society A*, 477(2253):20210162, 2021.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020a.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Andrew Stuart, Kaushik Bhattacharya, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33:6755–6766, 2020b.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, Anima Anandkumar, et al. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021.

Zongyi Li, Nikola Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, et al. Geometry-informed neural operator for large-scale 3d pdes. *Advances in Neural Information Processing Systems*, 36:35836–35854, 2023.

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, 2024.

Michael A Lieberman and Allan J Lichtenberg. Principles of Plasma Discharges and Materials Processing. *MRS Bulletin*, 30(12):899–901, 1994.

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

Richard H Macneal and Robert L Harder. A proposed standard set of problems to test finite element accuracy. *Finite elements in analysis and design*, 1(1):3–20, 1985.

Stefano Massaroli, Michael Poli, Jinkyoo Park, Atsushi Yamashita, and Hajime Asama. Dissecting neural odes. *Advances in neural information processing systems*, 33:3952–3963, 2020.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Md Ashiqur Rahman, Zachary E Ross, and Kamyar Azizzadenesheli. U-NO: U-shaped neural operators. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=j3oQF9coJd.

Md Ashiqur Rahman, Robert Joseph George, Mogab Elleithy, Daniel Leibovici, Zongyi Li, Boris Bonev, Colin White, Julius Berner, Raymond A Yeh, Jean Kossaifi, et al. Pretraining codomain attention neural operators for solving multiphysics PDEs. *Advances in Neural Information Processing Systems*, 37:104035–104064, 2024.

M Raissi, P Perdikaris, and GE Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

Michel Rappaz, Michel Bellet, Michel O Deville, and Ray Snyder. *Numerical modeling in materials science and engineering*, volume 20. Springer, 2003.

Shahid Rauf, Kallol Bera, Jason Kenney, and Prashanth Kothnur. Modeling of plasma processing reactors: review and perspective. *Journal of Micro/Nanopatterning, Materials, and Metrology*, 22(4):041503–041503, 2023.

Michał Jerzy Rewieński. *A trajectory piecewise-linear approach to model order reduction of nonlinear dynamical systems*. PhD thesis, Massachusetts Institute of Technology, Department of Electrical Engineering . . . , 2003.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in neural information processing systems*, 33:7462–7473, 2020.

Giovanni Stabile and Gianluigi Rozza. Finite volume POD-Galerkin stabilised reduced order methods for the parametrised incompressible Navier–Stokes equations. *Computers & Fluids*, 173: 273–284, 2018.

Shashank Subramanian, Peter Harrington, Kurt Keutzer, Wahid Bhimji, Dmitriy Morozov, Michael W Mahoney, and Amir Gholami. Towards foundation models for scientific machine learning: Characterizing scaling and transfer behavior. *Advances in Neural Information Processing Systems*, 36:71242–71262, 2023.

Floris Takens. Detecting strange attractors in turbulence. In *Dynamical Systems and Turbulence, Warwick 1980: proceedings of a symposium held at the University of Warwick 1979/80*, pp. 366–381. Springer, 2006.

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized Fourier neural operators. In *The Eleventh International Conference on Learning Representations*, 2023.

Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—An enhanced Fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

Xiongye Xiao, Defu Cao, Ruochen Yang, Gaurav Gupta, Gengshuo Liu, Chenzhong Yin, Radu Balan, and Paul Bogdan. Coupled multiwavelet neural operator learning for coupled partial differential equations. In *International Conference on Learning Representations*, 2023.

A DESIGN SPACES OF FNOS

A.1 DESCRIPTION ON EACH COMPONENT

Data space

(D1) Separate single-channel format. Each variable is treated as an independent function, $a(x) = (a^\alpha(x), a^\beta(x))$, and discretized separately as $\mathbf{a}^\alpha, \mathbf{a}^\beta \in \mathbb{R}^m$. This representation keeps the two components distinct throughout the architecture.

(D2) Two-channel format. The input is represented as a vector-valued function, $a : \mathcal{X} \to \mathbb{R}^2, \quad x \mapsto (a^\alpha(x), a^\beta(x))$, which in a discretized form becomes $\mathbf{a} = [\mathbf{a}^\alpha, \mathbf{a}^\beta] \in \mathbb{R}^{m\times 2}$. The output is represented analogously as $u(x) = [u^\alpha(x), u^\beta(x)]^\mathsf{T} \in \mathbb{R}^2$ and the discretized form $\mathbf{u} = [\mathbf{u}^\alpha, \mathbf{u}^\beta] \in \mathbb{R}^{m\times 2}$.

Lift operator, P

(P1) The lift operator $P : \mathbb{R}^1 \to \mathbb{R}^{d_v}$ and shared across the two input components, so that both variables are mapped into the latent space using the same transformation.

(P2) Alternatively, the lift operator can consist of two separate mappings, $P = (P^\alpha, P^\beta)$, each specific to one input component. This allows the two variables to have independent latent representations from the beginning.

(P3) Compatible with (D2), the lift operator is defined as $P : \mathbb{R}^2 \to \mathbb{R}^{d_v}$, mapping a two-channel input to a hidden state.

Fourier layers

(L1) The point-wise linear map is a single operator shared by the both variables.

(L2) The point-wise linear map is defined as two separate operators (W^α, W^β).

(G) For coupled systems, we define the global spectral convolution to perform coupling only in Fourier space. Specifically, each variable is first transformed independently, $\tilde{v}^\square(k) = \mathcal{F}v_\ell^\square(k)$, for $\square = \{\alpha, \beta\}$. Their spectral representations are then combined through a shallow encoder network $f^{\text{enc}}(\tilde{v}^\alpha(k), \tilde{v}^\beta(k))$, followed by the usual mode-wise kernel multiplication $\tilde{\tilde{v}}(k) = R_\phi(k)\tilde{v}(k)$. The result is decomposed into variable-specific coefficients using another shallow network, $(\tilde{\tilde{v}}^\alpha(k), \tilde{\tilde{v}}^\beta(k) = f^{\text{dec}}(\tilde{\tilde{v}}(k)$, and finally mapped back to the data space via the inverse Fourier transform, $v_{\ell+1}^\square(k) = \mathcal{F}^{-1}(\tilde{\tilde{v}}^\square(k))$, for $\square = \alpha, \beta$.

(G2) The standard global spectral convolution performs the Fourier transform, mode-wise kernel multiplication, followed by the inverse Fourier transform: $\mathcal{F}^{-1}\Big(R_\phi \cdot (\mathcal{F}v_\ell)\Big)(x)$.

Projection operator, Q

(Q1) The projection operator is shared across both output components, $Q : \mathbb{R}^{d_v} \to \mathbb{R}^1$.

(Q2) The projection operator is defined separately for each output component, $Q = (Q^\alpha, Q^\beta)$, with $Q^\square : \mathbb{R}^{d_v} \to \mathbb{R}^1$ for $\square \in \alpha, \beta$. From the adaptive-basis view perspective, Q can differ in whether the basis and the coefficients are shared or separate across the two variables:

- (Q2a) Shared basis and separate coefficients, and
- (Q2b) Separate basis and shared coefficients, and
- (Q2c) Separate basis and separate coefficients.

(Q3) Similar to (P3), this projection operator is defined as $Q : \mathbb{R}^{d_v} \to \mathbb{R}^2$, mapping a hidden state to a two-channel output.

A.2 RESULTING MODELS

Now we list realizations of FNOs obtained from some combinations of the above design choices.

- $\texttt{FNO}_{\texttt{m}}$: (D2) + (P3) + (L1) + (G2) + (Q3) - a two-channel input is processed by a standard FNO to produce a two-channel output.
- $\texttt{FNO}_{\texttt{c}}$: (D1) + (P1) + (L1) + (G2) + (Q1) - two single-channel variables are concatenated in a discrete representation, i.e., $[\mathbf{a}^{\alpha\mathsf{T}}, \mathbf{a}^{\beta\mathsf{T}}]^{\mathsf{T}} \in \mathbb{R}^{2m}$; the concatenated input is processed by a standard FNO to produce a single-channel vertically-concatenated output, i.e., $[\mathbf{u}^{\alpha\mathsf{T}}, \mathbf{u}^{\beta\mathsf{T}}]^{\mathsf{T}} \in \mathbb{R}^{2m}$.
- $\texttt{FNO}_{\texttt{x}}$: This is a class of the proposed models, which can be realized with the combinations of (D1) + (P1)/(P2) + (L1)/(L2) + (G) + (Q1)/(Q2). In the experimentation, with $\texttt{FNO}_{\texttt{x}}$, we refer to (D1) + (P2) + (L2) + (G) + (Q2c) unless otherwise stated.

B IMPLEMENTATION DETAILS

B.1 DESCRIPTIONS AND SPECIFICATIONS OF THE BASELINES AND THE PROPOSED MODELS

$\texttt{FNO}_{\texttt{m}}$ / $\texttt{FNO}_{\texttt{c}}$ / $\texttt{FNO}_{\texttt{x}}$ For all FNO-based baselines, we consider the same hyper-parameter 4 Fourier layers with the modes and width ($k = 12$ and $d_v = 20$). P is parameterized as a linear layer and Q is parameterized as a shallow MLP with one hidden layer. All nonlinearity functions are ReLU (Nair & Hinton, 2010). This fixed hyperparameter setup is to control the representational capacity contributed by these hyperparameters. Consequently, any observed performance gains can be attributed to the architectural modifications introduced in each variant rather than to differences in model capacity.

For defining the $\texttt{FNO}_{\texttt{x}}$, we consider the combination, (P1) + (L2) + (Q2c), which employs the shared lift operator, the separate point-wise linear maps, the proposed coupled global spectral convolution layer, and the separate sets of basis functions and separate sets of coefficients in the projection operator. The encoder/decoder, $f^{\text{enc}}/f^{\text{dec}}$, in the Fourier layers are set to simple linear layers. This choice is obtained from the ablation study.

$\texttt{pFNO}_{\texttt{c}}$ / $\texttt{pFNO}_{\texttt{x}}$ These parameterized variant takes the standard input (i.e., the input to $\texttt{FNO}_{\texttt{c}}$ and $\texttt{FNO}_{\texttt{x}}$), but augmented with the physical parameters, μ.

$\texttt{hpFNO}_{\texttt{c}}$ / $\texttt{hpFNO}_{\texttt{x}}$ These variants require one additional component, the hypernetwork, f^{hyper}. In our implementation, we consider a simple linear layer to parameterize the hypernework, and we model the hypernetwork to take (x, μ) along with the discrete reprsentation of the windowed history, $\{u(t-\tau)\}_{\tau=0}^{T_{\text{in}}-1}$. Thus, the hypernetwork becomes a local transform similar to the lift operator.

$\texttt{MWT}_{\texttt{c}}$ / CMWNO The multiwavelet-based operator learning framework (MWT) has been introduced as an approach to leverage multiwavelet expansions for differential equations. By decomposing solution operators into multiwavelet bases, MWT captures both global structures and localized variations in a flexible and efficient way, capable of handling multi-scale dynamics and heterogeneous patterns. The coupled extension of this has been introduced in the sequel (Xiao et al., 2023), which is named as $\texttt{MWT}_{\texttt{c}}$ and is designed to take vertically concatenated input variables.

Extending this idea, the coupled multiwavelet neural operator (CMWNO) adapts the MWT framework to systems of coupled partial differential equations. CMWNO introduces a strategy that decouples the operator kernels across different wavelet components during decomposition and reconstruction. This enables the model to efficiently learn and represent the interactions between multiple coupled fields while keeping computation tractable. Computationally, CMWNO consists of two distinct operators that communicate by exchanging hidden representations. A randomized scheduling mechanism, referred to as the 'dice' algorithm, determines the order in which these operators share information, enabling flexible and efficient coordination between the coupled components.

For both MWT and CMWNO, we use the implementation available from the Github repository[1]. The scaling parameter α is set to 12, the channel multiplier c is set to 16, the wavelet polynomial order c is set to 4, and the base polynomial basis is set to the 'Legendre' polynomials. The dice strategy

[1] https://github.com/joshuaxiao98/CMWNO/

is utilized as follows: for each batch we throw a random number from a uniform distribution [0,1] and if the sampled random number is below 0.5, the second operator takes the first operator's hidden representation through the roll-out. On the opposite case, the first operator takes the first operator's hidden representation through the roll-out.

CFNO `CFNO` mimics the operations of `CMWNO`, i.e., two separate operators enabling coupling effect via sharing a hidden representation of one operator to another. For `CFNO`, the backbone is `FNO` instead of `CFNO`. Also, the same dice approach described above is utilized.

Same as other FNO-based baselines, we consider the same hyper-parameter 4 Fourier layers with the modes and width ($k = 12$ and $d_v = 20$). P is parameterized as a linear layer and Q is parameterized as a shallow MLP with one hidden layer. All nonlinearity functions are ReLU. For `CFNO`s, we have two separate realizations of FNOs and in the third layer of `CFNO`s, the hidden representation from one `CFNO` is passed to another `CFNO`, where the direction is dictated by the dice approach.

DeepONets For the implementation of DeepONets (DONs), we base our implementation on a publicly available `PyTorch` version. [2] We modify the DONs to take multivariate input and produce multivariate output. We explore the combination of a (shared/separate) trunk net and a (shared/separate) branch net, and find that two separate branch networks and a shared trunk network perform the best. Each branch network takes the same input, which is concatenated time-windowed snapshots; then one branch network produces coefficients for the electron density and another branch network produces coefficients for the electric potential. That is,

$$n_{\mathrm{e}}(x, t+1) = \sum_{i=1}^{p} f_i^{\text{branch},n_{\mathrm{e}}}(\bar{a}_{\leq t}) f_i^{\text{trunk},n_{\mathrm{e}}}(x),$$

$$\phi(x, t+1) = \sum_{i=1}^{p} f_i^{\text{branch},\phi}(\bar{a}_{\leq t}) f_i^{\text{trunk},\phi}(x)$$

where $f^{\text{branch},n_{\mathrm{e}}}$ and $f^{\text{branch},\phi}$ denote two separate MLPs, taking time-windowed input

$$\bar{a}_{\leq t} = \begin{bmatrix} n_{\mathrm{e}}(x,t) & \cdots & n_{\mathrm{e}}(x, t - T_{\text{in}} + 1) \\ \phi(x,t) & \cdots & \phi(x, t - T_{\text{in}} + 1) \end{bmatrix},$$

where T_{in} decides the window length. The trunk net is modeled as a separate network, producing two separate sets of basis such that

$$[f^{\text{trunk},n_{\mathrm{e}}}(x), f^{\text{trunk},\phi}(x)] = [W^{n_{\mathrm{e}}} f^{\text{trunk}}(x) + b^{n_{\mathrm{e}}}, W^{\phi} f^{\text{trunk}}(x) + b^{n_\phi}],$$

where f^{trunk} denote a shared MLP, and $(W^{n_{\mathrm{e}}}, b^{n_{\mathrm{e}}})$ and (W^{ϕ}, b^{n_ϕ}) denote a set of model parameters specific to the electron density and electric potential, respectively.

For each branch networks, we use 4 hidden layers with 1024 neurons in each layer. For the trunk network, we use 3 hidden layers with 1024 neurons. For both network types, we consider ReLU for nonlinear activation.

U-Net For the implementation of U-Nets, we base our implementation on the code base we use for DeepONets. We modify the U-Net to take 1-dimensional and 2 channel inputs, and produce the output with the same dimensional specification. The U-Net architecture we consider has an encoder–decoder structure with skip connections.

The encoder progressively reduces the spatial resolution while increasing the feature dimension. It begins with an initial block (DoubleConv $\rightarrow$ ReLU), followed by four downsampling stages (MaxPool1d $\rightarrow$ DoubleConv $\rightarrow$ ReLU), where each stage halves the sequence length and expands the channel dimension. The spatial resolution is halved at every downsampling and the channel dimension is updated (32$\rightarrow$64$\rightarrow$64$\rightarrow$128$\rightarrow$128).

The decoder mirrors this process with four upsampling stages (Upsample $\rightarrow$ Concat (skip-connection) $\rightarrow$ DoubleConv $\rightarrow$ ReLU). At each stage, the feature map is upsampled by a factor of two, concatenated with the corresponding encoder feature (skip connection), and refined by a double convolution block. Finally, the output layer applies 1D convolution to project the features to the desired number of output channels.

[2] This is accessible through https://github.com/camlab-ethz/ConvolutionalNeuralOperator.

B.2 TRAINING DETAILS

All FNO variants and wavelet-based operators are trained for 500 epochs using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.0025, weight decay 0.0001, and mini-batches of size 10. For U-Nets, we consider the same hyperparameters, except for the total training epochs, which is set to 1,000, which is to mitigate the effect of the increased model sizes. For DeepONets, we again consider the same hyperparameters, except for the learning rate, which is set to 0.0001 and the total training epochs, which is set to 2,500 which is to compensate the decreased learning rate. For all baseline models, we repeat 5 different runs for varying random seeds.

B.3 ORTHOGONALITY LOSS

To measure the orthogonality of the learned basis functions, $\Psi(x) = [\psi_1(x), \ldots, \psi_{n_q}(x)]$, we define a discrete inner product over the spatial domain. For a spatial domain D and a measure $\rho(x)$, the continuous inner product is $\langle \psi_i, \psi_j \rangle = \int_D \psi_i(x)\psi_j(x)\mathrm{d}\rho(x)$. In practice, we approximate this integral using numerical quadrature. Specifically, we employ Gauss-type quadrature rules, which select quadrature nodes $x_k \in D$ and associated weights w_k, yielding $\langle \psi_i, \psi_j \rangle \approx \sum_{m=1}^{M} w_k \psi_i(x_k)\psi_j(x_k)$.

With this discrete formulation, we compute the Gram matrix, $[G]_{ij} = \langle \psi_i, \psi_j \rangle$, for $i, j = 1, \ldots, n_q$ and quantify deviation from orthonormality via $\mathcal{L}_{\text{ortho}} = \|G - I_{n_q}\|_{\text{F}}^2$, leading to the ultimate loss $\mathcal{L} = \mathcal{L}_{\text{mse}} + \lambda_{\text{ortho}}\mathcal{L}_{\text{ortho}}$. This approach allows for a principled enforcement of orthonormality in the learned basis while ensuring consistency with the underlying continuous inner product and measure. For λ_{ortho}, we consider $\{0.1, 0.01, 0.001\}$.

C 1D CCP: DESCRIPTION

C.1 GOVERNING EQUATIONS

The simplified one-dimensional capacitively coupled plasma physics is described by two equations: electron continuity equation and Poisson equation. The electron continuity equation describes the conservation of particles; electron density changes in time because of electron flows out at the surface and because they can be created/destroyed locally due to the chemical reactions. The Poisson equation relates how the electrostatic potential varies in space to the distribution of charges (i.e., electrons and ions) in the plasma. The governing equations are defined as follows:

$$\frac{\partial n_{\text{e}}}{\partial t} = -\frac{\partial \Gamma_{\text{e}}}{\partial x} + R, \qquad \text{(Electron continuity equation)}$$

$$\partial_{xx}\phi = \frac{e}{\epsilon_0}(n_{\text{e}} - n_{\text{io}}), \qquad \text{(Poisson equation)}$$

where $n_{\text{e}}(x,t)$ and $\phi(x,t)$ denote electron density and electric potential, which are the solutions of the equations, and $\partial_{\square}$ refers to a partial derivative with respect to $\square$. Γ_{e} is electron flux, defined as diffusion flux and drift flux such as

$$\Gamma_e = -D\partial_x n_{\text{e}} - \mu n_{\text{e}} \partial_x \phi$$

with electron diffusion coefficient D and electron mobility coefficient μ, which is defined as $D = eT_{\text{e}}/m_{\text{e}}\vartheta_m$. Here, $\mu = e/m_{\text{e}}\vartheta_m$. Here, e, T_{e}, m_{e}, and ϑ_m denote elementary charge, electron temperature, electron mass, and collision frequency. The ion density is defined as $n_{\text{io}} = R_0(x_2 - x_1)\sqrt{m_i/eT_{\text{e}}}$, where $R(x)$ and R_0 denote reaction rate and coefficient, respectively. The reaction rate is a spatially dependent quantity such that

$$R(x) = \begin{cases} R_0, & x \in [x_1, x_2] \cup [L - x_2, L - x_1] \\ 0, & \text{otherwise} \end{cases}.$$

The boundary conditions are specified as

$$n_{\text{e}} = 0, \phi = 0, \qquad \text{at } x = 0,$$
$$n_{\text{e}} = 0, \phi = V(t) \qquad \text{at } x = L,$$

where $V(t) = V_0 \sin(2\pi t)$. That is, the zero boundary condition is given to the electron density (at $x = 0, L$). The electric potential satisfies a homogeneous Dirichlet condition (at $x = 0$), and

a time-periodic Dirichlet condition at $x = L$. That is, the potential at the right boundary oscillates sinusoidally with amplitude V_0 and frequency f.

Table 4 lists up the important parameters, their description, and their values. Other important constants are elementary charge, $e = 1.6 \times 10^{-19}$ (C) and vacuum permittivity $\epsilon_0 = 8.854 \times 10^{-12}$ ($C^2 kg^{-1} m^{-3} s^2$).

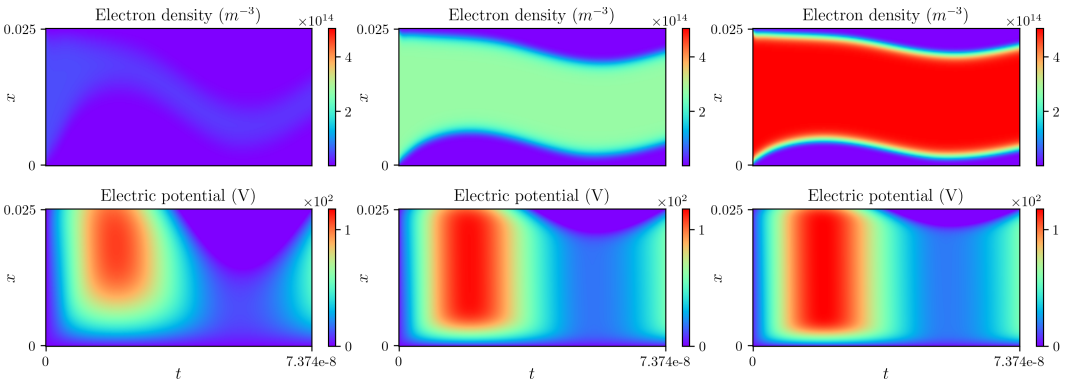
Table 4: Input parameter symbols, descriptions, and values

Symbols	Descriptions	Values (unit)
L	Domain Length	0.025 (m)
x_1, x_2	Reaction area	0.005(m), 0.01(m)
f	Driving frequency	13.56 (MHz)
V_0	Driving voltage amplitude	[100, 300] (V)
R_0	Reaction rage coefficients	[2.7×10^{19}, 2.7×10^{20}‘] ($m^{-3}s^{-1}$)
T_e	Electron temperature	3 (eV)
m_i	Ion mass	[1.67×10^{-26}, 6.68×10^{-26}] (kg)
m_e	Electron mass	9.109×10^{-31} (kg)
ϑ_m	Collision frequency	10^8 (s^{-1})

C.2 Visualization of the solution snapshots

To provide better understanding on the dynamics of the benchmark problem, we present some collections of solution snapshots for varying input parameters, reaction rate (Figure 5) and driving voltage (Figure 6) in the first RF cycle. All Figures presents the two spatio-temporal fields, the electron density $n_e(x,t)$ and the electric potential $\phi(x,t)$, on a $t-x$ plane, in a heatmap format.

Figure 5 shows the solution snapshots for varying reaction rates $R_0 = \{2.7 \times 10^{19}, 1.485 \times 10^{20}, 2.7 \times 10^{20}\} m^{-}3s^{-1}$. The reaction rate serves as a quantity that determines how fast ionization occurs. The reaction rate drives the plasma density and sustainment in a CCP discharge. The higher the reaction rates, the faster the ionization is, and the higher plasma density is. Reaction rate can be controlled using power in plasma processing systems. Figure 5 shows some example solution snapshots for varying reaction rate; from the left panel to the right panel, the reaction rate increases, which results in higher electron density and thinner sheaths.

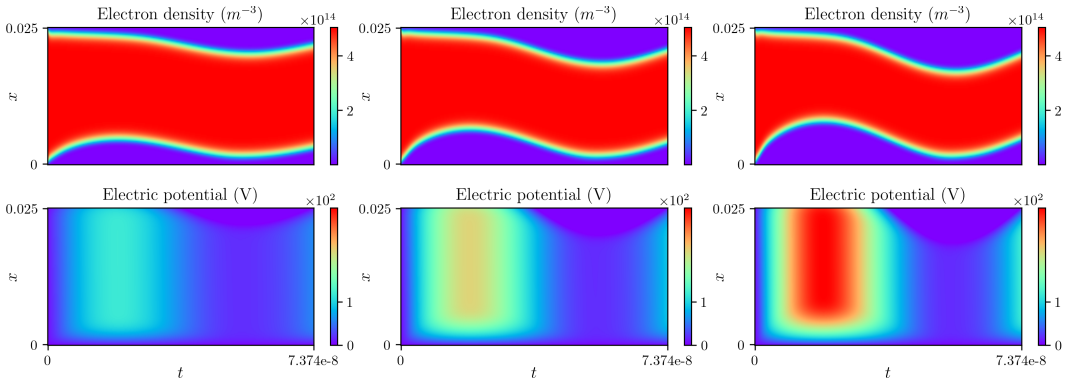


(a) $R_0 = 2.7 \times 10^{19}$ (b) $R_0 = 1.485 \times 10^{20}$ (c) $R_0 = 2.7 \times 10^{20}$

Figure 5: [1D CCP] Solution snapshots presented in heatmap; the solutions are collected for varying reaction rates $R_0 = \{2.7 \times 10^{19}, 1.485 \times 10^{20}, 2.7 \times 10^{20}\}$.

Figure 6 shows the solution snapshots for varying driving voltages $V_0 = \{100, 200, 300\}$ in the first RF cycle. The driving voltage, given as the boundary condition, establishes the time-varying

electric potential fields. The higher the driving voltage, the larger is the sheath voltage, resulting in stronger electric fields. The sheath is thicker at higher electric field. Figure 6 shows example solution snapshots for varying driving voltage: (a) $V_0 = 100$, (b) $V_0 = 200$, and (c) $V_0 = 300$.



(a) $V_0 = 100$ (b) $V_0 = 200$ (c) $V_0 = 300$

Figure 6: [1D CCP] Solution snapshots presented in heatmap; the solutions are collected for varying driving voltages $V_0 = \{100, 200, 300\}$.

Likewise, varying the ion-mass also provides variations in the dynamics. However, compared to other two physical parameters, the reaction rate and the driving voltage, changing the ion-mass parameter has slightly less physical impact on the dynamics and thus, the solution snapshots.

D 1D CCP: Additional experimental details and results

D.1 1D CCP results

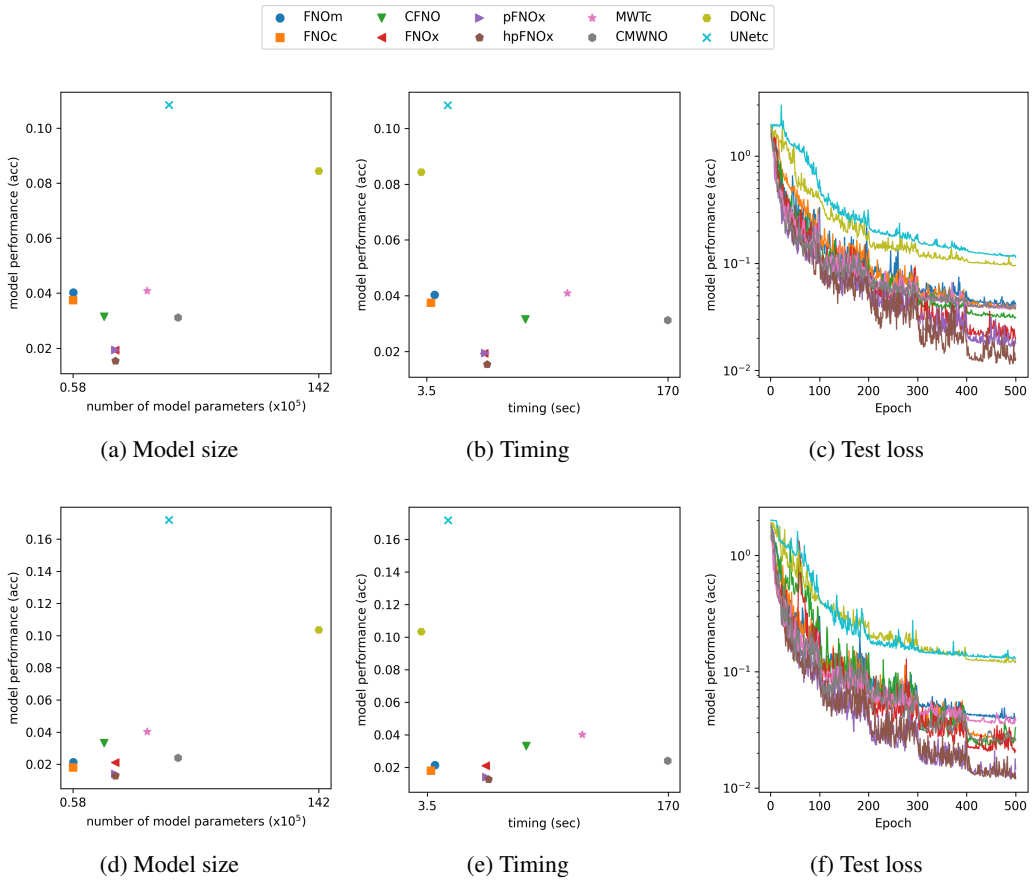
In the following sections, we present additional experimental results that are not presented in the main body due to the space limit. The additional experiments include an ablation study on different design choices of FNOs, a study on effectiveness of the proposed hypernetwork and modulation based architecture for the parametric extension, and a study on extrapolation in the parameter space.

D.1.1 Additional results

Continuing from the experimental results presented in the main text (i.e., the "1-dimensional parameter space" experimentation), we present the further detailed information regarding the computations: the model sizes and the computational timings, depicted in Figure 7. This time, the information obtained from the experimentation with the datasets that vary the reaction rate and the ion masses. Again, the model size is measured in the number of trainable model parameters and the timing reports a per-epoch training time measured and averaged across the total number of epochs. We can make the similar observations here: the proposed methods $\texttt{FNO}_\texttt{x}$, $\texttt{pFNO}_\texttt{x}$, and $\texttt{hpFNO}_\texttt{x}$ achieve improvement in accuracy without sacrificing the model size and the computational wall time. In case of the ion-mass scenario, we see relatively small improvements compared to other two scenarios (the reaction rates and the driving voltage). This is largely due to the fact that varying the ion-mass provide less changing physical dynamics and taking the windowed history input, $\{u(t-\tau)\}_{\tau=0}^{T_{\text{in}}-1}$, allows other non-parameterized NOs to infer those physical parameters to some extent. Still, the proposed methods $\texttt{FNO}_\texttt{x}$, $\texttt{pFNO}_\texttt{x}$, and $\texttt{hpFNO}_\texttt{x}$ improves the performance in that scenario while minimally affecting the model size and the training time.

D.1.2 Ablation on coupled extension

To investigate the effectiveness of each design component, we perform an ablation study on several different combinations of the design choices. We begin with the *default* combination, Ⓟ + Ⓛ + Ⓠ, which consists of the shared lift/projection operators and the point-wise linear map in the Fourier



(a) Model size (b) Timing (c) Test loss

(d) Model size (e) Timing (f) Test loss

Figure 7: [1D CCP] Plots depict the number of model parameters versus the model performance (left), computational timing versus the model performance (right), for varying reaction rates R_0 (top) and ion masses m_i (bottom)

layers. We first consider replacing the point-wise linear map in the Fourier layer to be a set of separate point-wise linear maps, W^α and W^β. After that, we investigate the projection operator by testing three different combinations: Shared/separate basis functions Ψ and shared/separate coefficients Ξ. Finally, we test the effect of a set of separate lift operators. Table 5 reports the results of this ablation study. The normalization shown in the last entries of the table is layer norm applied to the pre-activation values in the projection operator (i.e., $W_1 v_T(x) + b_1$). From this ablation study, we choose the combination, (P1) + (L2) + (Q2c) with layer norm, as $\texttt{FNO}_\text{x}$ for all experimentation.

Table 5: Performance comparisons of different realizations of $\texttt{FNO}_\text{x}$. The models are trained for 1-dimensional parameter space, (reaction rate). The performance is measured in the relative ℓ^2-error (nRMSE); the reported numerical values refer to the mean ($\pm$ std. dev).

Variant	Description	Relative errors
(P1) + (L1) + (Q1)	Default setup	0.0341 ($\pm$ 0.0044)
(P1) + (L2) + (Q1)	Separate W	0.0281 ($\pm$ 0.0048)
(P1) + (L2) + (Q2a)	Separate W and coefficients Ξ	0.0259 ($\pm$ 0.0032)
(P1) + (L2) + (Q2b)	Separate W and basis Ψ	0.0315 ($\pm$ 0.0028)
(P1) + (L2) + (Q2c)	Separate W and basis Ψ and coefficients Ξ	0.0275 ($\pm$ 0.0062)
(P1) + (L2) + (Q2b) w. norm	Separate W and coefficients Ξ with layer norm	0.0205 ($\pm$ 0.0043)
(P1) + (L2) + (Q2c) w. norm	Separate W and basis Ψ and coefficients Ξ with layer norm	0.0193 ($\pm$ 0.0059)
(P2) + (L2) + (Q2c) w. norm	Separate P and W and basis Ψ and coefficients Ξ with layer norm	0.0204 ($\pm$ 0.0035)

D.1.3 PARAMETRIC EXTENSION

Study on the effectiveness of the proposed model architecture To further investigate the effectiveness of the proposed parameteric extension, we compare the performances of the non-parameterized versions, $\texttt{FNO}_\texttt{c}$ and $\texttt{FNO}_\texttt{x}$ against their own two parameterized versions, ($\texttt{pFNO}_\texttt{c}$, $\texttt{hpFNO}_\texttt{c}$) and ($\texttt{pFNO}_\texttt{x}$, $\texttt{hpFNO}_\texttt{x}$). Table 6 reports the models' performance measured in all three physical parameter scenarios. Taking the non-parameterized version as the baseline, Table 6 reports the achieved prediction accuracy of each model and their improvements over the non-parameterized versions, presented in the percentage. The results indicate that simply augmenting physical parameters to input provide some improvements in prediction accuracy. However, the new architectural modification (i.e., hypernetwork and modulation) brings significantly improved results (upto 68.01% improvement).

Table 6: Performance comparisons of non-parameterized ($\texttt{FNO}_\texttt{c}$, $\texttt{FNO}_\texttt{x}$) and parameterized ($\texttt{pFNO}_\texttt{c}$, $\texttt{hpFNO}_\texttt{c}$, $\texttt{pFNO}_\texttt{x}$, $\texttt{hpFNO}_\texttt{x}$) models. All models are tested on three scalar parameters (reaction rate, driving voltage, and ion mass) individually. The performance is measured in the relative ℓ^2-error; the reported numerical values refer to the mean ($\pm$ standard deviation, improvement over non-parameterized version).

Model	Reaction rate	Driving voltage	Ion mass
$\texttt{FNO}_\texttt{c}$	0.0375 ($\pm$ 0.0055)	0.0873 ($\pm$ 0.0200)	0.0299 ($\pm$ 0.0030)
$\texttt{pFNO}_\texttt{c}$	0.0334 ($\pm$ 0.0101, 10.94 %)	0.0454 ($\pm$ 0.0094, 47.92 %)	0.0284 ($\pm$ 0.0023, 5.08 %)
$\texttt{hpFNO}_\texttt{c}$	0.0196 ($\pm$ 0.0021, 47.57 %)	0.0279 ($\pm$ 0.0016, 68.01 %)	0.0210 ($\pm$ 0.0036, 29.84 %)
$\texttt{FNO}_\texttt{x}$	0.0193 ($\pm$ 0.0059)	0.0345 ($\pm$ 0.0108)	0.0212 ($\pm$ 0.0062)
$\texttt{pFNO}_\texttt{x}$	0.0194 ($\pm$ 0.0075, -0.51 %)	0.0278 ($\pm$ 0.0053, 19.30 %)	0.0142 ($\pm$ 0.0021, 32.91 %)
$\texttt{hpFNO}_\texttt{x}$	**0.0154** ($\pm$ 0.0029, 20.12 %)	**0.0192** ($\pm$ 0.0040, 44.40 %)	**0.0128** ($\pm$ 0.0017, 39.83 %)

Study on the out-of-distribution samples In the main experiments, we follow the standard FNO testing strategy. In our parameterized scenarios, however, we can explicitly sample parameters that lie entirely outside the training range, which we define as out-of-distribution (OOD) testing. As a benchmark case, we vary the reaction rate R_0. For training, 101 equidistance values of the reaction rate is sampled from [2.7×10^{19}, 2.7×10^{20}]; for OOD testing, the interval is expanded outward by 0.2×10^{19} in both directions. Table 7 reports the performance of all considered models on ID test samples and OOD test samples. Overall, the performance measured on OOD produces increased numbers, but the trend across the compared methods remains the same. The proposed $\texttt{FNO}_\texttt{x}$ model family produce more accurate predictions than the existing methods.

Table 7: Performance comparisons of non-parameterized ($\texttt{FNOc}$, $\texttt{FNO}_\texttt{x}$) and parameterized ($\texttt{pFNOc}$, $\texttt{hpFNOc}$, $\texttt{pFNO}_\texttt{x}$, $\texttt{hpFNO}_\texttt{x}$) models for OOD samples (reaction rate). The performance is measured in the relative ℓ^2-error; the reported numerical values refer to the mean ($\pm$ standard deviation, improvement over non-parameterized version).

Model	ID	OOD
$\texttt{FNO}_\texttt{m}$	0.0403 ($\pm$ 0.0012)	0.0628 ($\pm$ 0.0075)
$\texttt{FNO}_\texttt{c}$	0.0375 ($\pm$ 0.0055)	0.0673 ($\pm$ 0.0097)
$\texttt{pFNO}_\texttt{c}$	0.0334 ($\pm$ 0.0101)	0.0498 ($\pm$ 0.0069)
$\texttt{hpFNO}_\texttt{c}$	0.0196 ($\pm$ 0.0021)	0.0374 ($\pm$ 0.0051)
$\texttt{FNO}_\texttt{x}$	0.0193 ($\pm$ 0.0059)	0.0446 ($\pm$ 0.0094)
$\texttt{pFNO}_\texttt{x}$	0.0194 ($\pm$ 0.0075)	0.0343 ($\pm$ 0.0059)
$\texttt{hpFNO}_\texttt{x}$	**0.0154** ($\pm$ 0.0029)	**0.0303** ($\pm$ 0.0050)

E GRAY–SCOTT EQUATIONS

The Gray–Scott equations form a system of coupled reaction-diffusion equations that describe the spatiotemporal dynamics of interacting chemical species. This model can reproduce a rich variety of natural patterns, including structure resembling bacterial colonies, spiral waves, and coral-like formations. In the system, two state variables u and v each undergo independent diffusion and linear growth/decay, while their interaction is governed by the nonlinear reaction term.

The governing equations are defined as follows:

$$\frac{\partial u}{\partial t} = \epsilon_1 \frac{\partial^2 u}{\partial x^2} + F(1-u) - \lambda_1 u v^2,$$
$$\frac{\partial v}{\partial t} = \epsilon_2 \frac{\partial^2 v}{\partial x^2} - (K+F)v + \lambda_2 u v^2,$$

where $u(x,t)$ denotes concentration of the feed species, which is continuously supplied to the system and $v(x,t)$ denotes concentration of the reactant species, which is produced and consumed through interaction with u. The diffusion coefficients of u and v are denoted as ϵ_1 and ϵ_2, respectively. Other parameters include: F, the feed rate, representing the rate at which species u is introduced into the system from an external source, K, he decay rate of species v, which accounts for natural removal, and λ, the reaction rate constant associated with the nonlinear term $-\lambda uv^2$, which describes the autocatalytic process where one unit of u reacts with two units of v.

E.1 EXPERIMENTATION

In the experiment, we consider the parameterized dynamics realized by varying the diffusion coefficient $\epsilon_1 \in [0.1, 10]$ and varying the feed rate $F \in [0.1, 10]$. We collect 101 samples of equi-distanced coefficient/feed rate values while the other parameters are fixed as $\epsilon_2 = 0.05$, $K = 0.1$ and $(\lambda_1, \lambda_2) = (2, 5)$. For sampling an initial condition, we follow the approach described in (Xiao et al., 2023), sampling $u(x,0)$ using the smooth random functions in the `Chebfun` package (Driscoll et al., 2014) and sampling $v(x,0)$ from Gaussian random field, $v(x,0) \sim \mathcal{N}(0, 7^4(-2\pi + 7^2 I)^{-2.75})$. For collecting trajectories for constructing training/test datasets, initial value problems are numerically solved using a fourth-order time-integrator, exponential time differencing fourth-order Runge–Kutta (ETDRK4), utilizing the chebfun software. We base the implementation the publicly available `Matlab` script[3], which is the implementaion accompanyed by the research work (Xiao et al., 2023). The IVPs are solved with a spatial resolution, 1024, and then the collected solution snapshots are subsampled in the spatial domain, leading to the final spatial resolution, 128. The time integration is performed for the time interval [0,0.1] with 31 time steps.

Table 8: Performance comparisons between the proposed methods and the baseline methods. All models are tested on a scalar parameter: the diffusion coefficient. The performance is measured in the relative ℓ^2-error; the reported numerical values refer to the mean ($\pm$ standard deviation)

Model	Diffusion coeff, ϵ_1	Feed rate, F
$\texttt{FNO}_\texttt{m}$	0.0129 ($\pm$ 0.0032)	0.0116 ($\pm$ 0.0026)
$\texttt{FNO}_\texttt{c}$	0.0089 ($\pm$ 0.0052)	0.0097 ($\pm$ 0.0030)
CFNO	0.0093 ($\pm$ 0.0016)	0.0161 ($\pm$ 0.0045)
$\texttt{MWT}_\texttt{c}$	0.0055 ($\pm$ 0.0011)	0.0092 ($\pm$ 0.0021)
CWMNO	0.0048 ($\pm$ 0.0015)	0.0293 ($\pm$ 0.0078)
$\texttt{DON}_\texttt{c}$	0.0243 ($\pm$ 0.0029)	0.0138 ($\pm$ 0.0020)
$\texttt{U-Net}_\texttt{c}$	0.0239 ($\pm$ 0.0160)	0.0159 ($\pm$ 0.0071)
$\texttt{FNO}_\texttt{x}$	0.0039 ($\pm$ 0.0002)	0.0075 ($\pm$ 0.0016)
$\texttt{pFNO}_\texttt{x}$	0.0027 ($\pm$ 0.0008)	0.0041 ($\pm$ 0.0010)
$\texttt{hpFNO}_\texttt{x}$	0.0022 ($\pm$ 0.0010)	0.0022 ($\pm$ 0.0006)

[3]This script is available in https://github.com/joshuaxiao98/CMWNO/.

Table 8 reports the prediction accuracy of all considered method. The results provide similar observations with those of the 1D CCP experimentation; the proposed methods, the parameterized variants of $\texttt{FNO}_{\texttt{x}}$ achieve the lowest errors compared to existing baselines with 54% (for the diffusion coefficient case) and 72% (for the feed rate case) improvements. The coupled extension of FNO, $\texttt{FNO}_{\texttt{x}}$, improves the performance compared to other baselines. This is achieved without significantly scarifying the training/inference time and the memory footprint compared to $\texttt{FNO}_{\texttt{m}}$ or $\texttt{FNO}_{\texttt{c}}$ while other baselines require significantly larger-sized models and/or computational times. In the varying feed rate scenario, the approaches leveraging two separate operators struggle further to capture the accurate dynamics, resulting in higher prediction errors.