ACROSS RF CYCLES: POISSON-CLOSED NEURAL OPERATORS FOR CAPACITIVELY COUPLED PLASMAS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neural operators (NOs) offer efficient surrogates for time-dependent physical systems, but their use in specialized plasma applications remains limited. We develop a tailored NO for parameterized one-dimensional capacitively coupled plasmas (CCPs), a discharge widely used for plasma etching and deposition in semiconductor manufacturing. Under radio-frequency (RF) excitation, the potential and the electron density near the electrodes respond within each RF cycle, while the plasma density and ion flow evolve slowly over many cycles toward an RF-periodic steady state. These multiscale, long-horizon dynamics are difficult for conventional autoregressive NOs because prediction errors accumulate across repeated RF cycles. We address these challenges through a Poisson-closure rollout, Floquet–Poincaré-inspired regularization, and random subsequence training for scalable long-trajectory learning. We introduce a parameterized 1D CCP fluid model as a benchmark for developing and evaluating NOs.

1 INTRODUCTION

Neural operators (NOs) are a class of models designed to learn mappings between function spaces, a formulation that naturally arises in forward problems governed by parameterized partial differential equations (PDEs) (Lu et al., 2021; Kovachki et al., 2023; Boullé & Townsend, 2024; Azizzadenesheli et al., 2024). Since their introduction, numerous NO architectures have demonstrated strong predictive performance across a broad range of canonical PDE benchmarks (Li et al., 2021; Tran et al., 2023; Li et al., 2023; Rahman et al., 2024; Buitrago Ruiz et al., 2025). While these benchmarks have been instrumental in establishing the general capabilities of operator learning, the adoption of NOs in specialized scientific and engineering applications remains comparatively limited.

One such domain is plasma physics, where NOs have seen limited adoption, particularly for capacitively coupled plasmas (CCPs). CCPs are the workhorse discharge for plasma etching and deposition in semiconductor manufacturing (Lieberman & Lichtenberg, 1994; Chabert & Braithwaite, 2011; Rauf et al., 2023) and polymer surface modification (Liston et al., 1993). Existing neural-operator studies in plasma physics have primarily applied general-purpose FNO architectures to fusion-plasma surrogate modeling (Gopakumar et al., 2024; Carey et al., 2025), while recent work has begun to tailor them to the coupled and parameterized structure of CCP dynamics (Jing et al., 2026).

Nevertheless, important challenges still remain: the evolution equations are subject to a Poisson constraint, the system must be propagated over many radio-frequency (RF) cycles to approach a quasi-steady periodic regime, and this RF boundary condition induces rapidly oscillating solution profiles. In our setting, capturing these dynamics requires autoregressive rollouts of $\geq$ 6,000 time steps, amplifying both computational cost and error accumulation. These challenges motivate a more specialized NO framework for accurate and tractable long-horizon prediction of CCP dynamics.

To address these challenges, we develop a CCP-tailored NO framework with three complementary components. First, we introduce a Poisson-closure rollout, in which the NO advances the evolutionary variables, while the electric potential is recovered at every time step by numerically solving Poisson's equation with the prescribed RF boundary condition. This hybrid decomposition explicitly enforces the elliptic constraint and avoids requiring the NO to directly predict the strongly oscillatory potential.

Second, we introduce Floquet–Poincaré-inspired (FP) regularization to exploit the periodic structure induced by the RF forcing. An auxiliary cycle-to-cycle NO predicts the state one RF period ahead

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

and provides phase-aligned supervision to the local rollout, thereby reducing drift accumulated across repeated RF cycles. The auxiliary operator is used only during training, leaving the inference cost of the backbone NO model unchanged.

Finally, we adopt random temporal subsequence training, optimizing the model over short contiguous windows sampled from the full trajectories. This strategy reduces the cost of training for long-horizon prediction while exposing the model to diverse transient regimes and phases of the RF cycle. Together, these components enable physics-consistent and scalable autoregressive prediction of CCP dynamics.

Our contributions are summarized as follows:

- We introduce a Poisson-closure rollout for explicit enforcement of the elliptic constraint.
- We propose Floquet–Poincaré-inspired regularization to mitigate drift across RF cycles.
- We systematically study random-subsequence training for scalable long-trajectory learning.
- We introduce a parameterized 1D CCP benchmark with $> 6{,}000$-step autoregressive rollouts.

2 Problem description: 1D Capacitively Coupled Plasma Physics

In this section, we begin by introducing a problem of interest: 1-dimensional CCP model, describing a low-temperature plasma generated between two electrodes under an applied alternating voltage, where the system operates analogously to a capacitor. Figure 1 depicts the spatial domain and the boundary conditions, in particular, the voltage given as the sinusoidal RF signal.

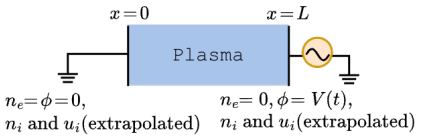


Figure 1: The spatial domain and boundary conditions.

In this formulation, both electrons (e) and ions (i) are treated as fluid species, with electron transport modeled using a drift-diffusion approximation (Eqs. (1) and (5)) and ion dynamics governed by a momentum-based fluid equation (Eqs. (2)–(3)). These transport equations are self-consistently coupled with Poisson's equation (Eq. (4)) to resolve the electric potential arising from charge separation. The resulting system captures the interaction between charged particle transport and the oscillating electric field, enabling the modeling of key plasma phenomena such as sheath formation, spatiotemporal variation in charge density, and nonlinear coupling between species. This framework provides a physically grounded description suitable for analyzing plasma behavior in applications including semiconductor processing and surface modification.

In equations, the above-described phenomena are governed by a set of four coupled partial differential equations, where we use the shorthand notation $\partial_t = \partial/\partial t$, $\partial_x = \partial/\partial x$, and $\partial_{xx} = \partial^2/\partial x^2$:

$$\partial_t n_\mathrm{e} + \partial_x \Gamma_\mathrm{e} = R, \qquad \text{(Electron continuity equation)} \tag{1}$$

$$\partial_t n_\mathrm{i} + \partial_x (n_\mathrm{i} u_\mathrm{i}) = R, \qquad \text{(Ion continuity equation)} \tag{2}$$

$$m_\mathrm{i}\partial_t u_\mathrm{i} + m_\mathrm{i} u_\mathrm{i}\partial_x u_\mathrm{i} = -q\partial_x \phi - m_\mathrm{i}\nu_\mathrm{i} u_\mathrm{i}, \qquad \text{(Ion momentum equation)} \tag{3}$$

$$\partial_{xx}\phi = -e/\epsilon_0 (n_\mathrm{e} - n_\mathrm{i}), \qquad \text{(Poisson equation),} \tag{4}$$

where $n_\mathrm{e}(x,t)$, $n_\mathrm{i}(x,t)$, $u_\mathrm{i}(x,t)$, and $\phi(x,t)$ denote the electron density, ion density, ion velocity, and electric potential, respectively. Physical parameters include the reaction rate $R(x)$, ion mass m_i, ion collision frequency ν_i, elementary charge e, and the vacuum permittivity ϵ_0. Lastly, the electron flux is defined as a sum of diffusion flux and drift flux such that

$$\Gamma_\mathrm{e} = -D\partial_x n_\mathrm{e} - \mu n_\mathrm{e}\partial_x \phi, \qquad \text{(Electron Flux)} \tag{5}$$

where D and μ denote electron diffusion coefficient and electron mobility coefficient.

The CCP system is driven by the time-periodic boundary condition $\phi(L,t) = V_0 \sin(2\pi f_\mathrm{RF} t)$, which defines the RF period $T_\mathrm{RF} = 1/f_\mathrm{RF}$, leading to non-autonomous, but periodic in time governing dynamics. Figure 2 shows an instance of resulting solution trajectories; the trajectories exhibit two distinct temporal scales: rapid oscillations within each RF cycle and a slower relaxation of the cycle-wise solution profiles toward a quasi-periodic regime. The rapid oscillations appear in all physical species, but the electric potential (ϕ in Figure 2 (a)), whose dynamics is directly governed the boundary condition, exhibits the noticeable oscillatory behavior. Further details on the governing equations and their physical interpretation are provided in Appendix A.1.

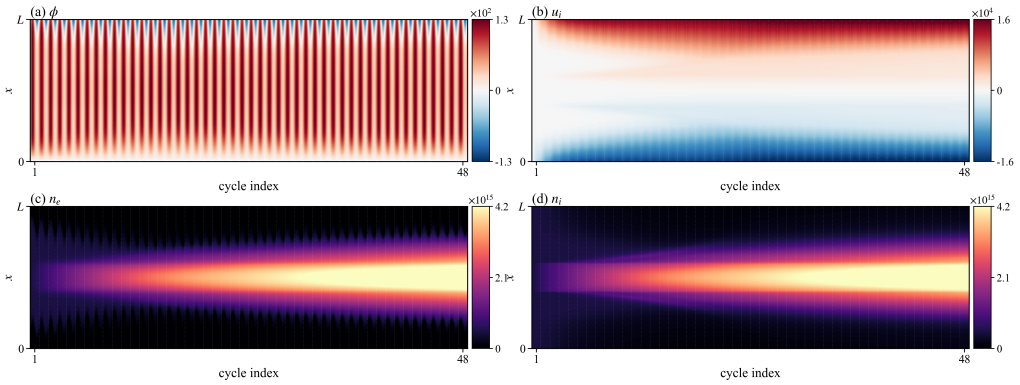


Figure 2: An example solution profile: the solution obtained at $R_0 = 2 \times 10^{21}$ via numerical simulation. The figure shows simulation run of 48 RF cycles (horizontal axis); (a) electric potential $\phi(x, t)$ demonstrate periodic behavior dictated by the boundary condition, (b) ion velocity u_i also exhibits convergent behavior, (c) and (d) electron and ion densities n_{e} and n_{i} exhibit large numerical scales compared to other two physical variables and convergent behavior as the simulation proceeds.

Key properties of the system This CCP system presents several challenges for neural surrogate modeling: (1) *nonlinearly coupled dynamics*, arising from the interaction between charged species and the electric field; (2) *parametric dependence on physical coefficients*, leading to variability across operating regimes; in particular, dependence on the source amplitude R_0, which spans three decades: at the low end, pair production over the simulated window is small compared with the initial density and the plasma decays from its initial state; at the high end, the density grows by up to an order of magnitude, with correspondingly thinner sheaths, (3) a *coupled differential–algebraic structure*, in which evolution equations (Eqs. (1)–(3)) are constrained by the Poisson equation (Eq. 4) for the electric potential, (4) a *long-rollout simulation* to reach quasi-periodic state (e.g., Figure 2 contains 48 RF cycles with 256 time-steps in each RF) and (5) *two-temporal-scale evolution behaviors*, short-term oscillation driven by the RF boundary condition and long-term behavior converging to quasi-periodic steady-state.

3 Neural Operators

NOs are data-driven surrogate models that aim to learn mappings between infinite-dimensional function spaces, rather than conventional finite-dimensional input–output relationships. This property makes them well-suited for approximating solution operators of PDEs (Lu et al., 2021; Kovachki et al., 2023; Boullé & Townsend, 2024; Azizzadenesheli et al., 2024). By operating on discretized representations of continuous functions, NOs can efficiently approximate complex physical systems while maintaining consistency with their underlying functional structure.

Let $\mathcal{A}$ and $\mathcal{U}$ denote function spaces, and consider a nonlinear operator $G : \mathcal{A} \to \mathcal{U}$. Neural operators seek to approximate G with a parameterized model G_Θ drawn from a suitable neural network hypothesis class. Given training samples $\{(a^i, u^i)\}_{i=1}^n$, where $a^i \in \mathcal{A}$ and $u^i = G(a^i) \in \mathcal{U}$, the model parameters Θ are learned by minimizing a supervised loss of the form $\min_\Theta \sum_{i=1}^n L\big(G_\Theta(a^i), u^i\big)$.

3.1 Fourier Neural Operators

Fourier neural operators (FNOs) are a class of neural operators that parameterize nonlinear integral operators and evaluate them efficiently in the Fourier domain. Specifically, FNOs define mappings $G_\Theta : \mathcal{A} \to \mathcal{U}$ by lifting the input function into a latent representation, followed by a sequence of Fourier layers. The input $a(x)$ is first projected into a higher-dimensional feature space via a local transformation $v_0(x) = P(a(x))$, where P is typically a linear mapping. The latent representation is then updated iteratively through layers of the form

$$v_{\ell+1}(x) = \sigma\Big(W v_\ell(x) + \big(\mathcal{K}(a;\phi)v_\ell\big)(x)\Big), \qquad \forall x \in D, \tag{6}$$

where W is a learnable linear transformation, and $\mathcal{K}$ denotes a kernel integral operator.

The operator $\mathcal{K}$ is defined as

$$\big(\mathcal{K}(a;\phi)v_\ell\big)(x) = \int_D \kappa(x, y, a(x), a(y); \phi)\, v_\ell(y)\, \mathrm{d}y, \qquad \forall x \in D,$$

where κ is a parameterized kernel with learnable parameters ϕ. To enable efficient computation, FNOs implement this in Fourier space. The integral operator is approximated via convolution, yielding

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

where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier and inverse Fourier transforms, respectively, and R_ϕ represents the Fourier-domain parameterization of the kernel.

Finally, the latent representation after T layers is projected back to the output space via a local mapping $u(x) = Q(v_T(x))$, where Q is typically a shallow multilayer perceptron.

3.2 Coupled and Parameterized Fourier Neural Operators

Recent work (Jing et al., 2026) has extended FNOs ($\texttt{FNO}_\text{x}$) to address more complex settings, including coupled systems and parameterized dynamics. In particular, parameterized dynamics can be effectively handled using a hypernetwork-augmented FNO ($\texttt{hpFNO}_\text{x}$), which incorporates physical parameters into the operator learning process.

Coupled FNOs The extension handles coupled systems involving multiple interdependent fields. For K physical fields, the objective is to learn a joint operator $(u^1, \ldots, u^K) = G_\Theta(a^1, \ldots, a^K)$, where each component evolves under shared dynamics while interacting with the others.

A representative formulation introduces coupling within the FNO architecture ($\texttt{FNO}_\text{x}$) while preserving its core computational structure. In particular, interactions across variables are modeled in the spectral domain, where their Fourier representations are combined through a lightweight mixing mechanism. Specifically, each component is first transformed independently into Fourier space, $\hat{v}_\ell^k = \mathcal{F}(v_\ell^k)$ for $k = 1, \ldots, K$, and concatenated into a joint representation. This combined representation is then processed by a shallow encoder $\hat{v}_\ell = f^{\text{enc}}([\hat{v}_\ell^1, \ldots, \hat{v}_\ell^K])$, followed by a mode-wise kernel operation $\tilde{v}_\ell = R_\phi \hat{v}_\ell$. The result is mapped back to the spatial domain via $v_\ell^{\text{mix}} = \mathcal{F}^{-1}(\tilde{v}_\ell)$ and subsequently decomposed into component-wise representations through a decoder $[\hat{v}_\ell^1, \ldots, \hat{v}_\ell^K] = f^{\text{dec}}(v_\ell^{\text{mix}})$.

$\texttt{FNO}_\text{x}$ employs a shared lift operator to map all input components into a common latent space. The Fourier layers then apply variable-specific point-wise linear transformations (i.e., W in Eq. (6)) together with a shared spectral convolution that introduces cross-variable interactions. Finally, the outputs are reconstructed using separate projection operators with distinct adaptive basis functions and coefficients for each component. This combination of shared and component-wise operators allows the model to capture both common structure and variable-specific behavior in coupled systems.

Parameterized Fourier Neural Operators In $\texttt{hpFNO}_\text{x}$, a compact hypernetwork (Ha et al., 2017) is introduced to encode physical parameters $\mu \in \mathbb{R}^{n_\mu}$. As depicted in Figure 3, the hypernetwork takes μ (together with spatial coordinates and optionally an input function) as input and produces a collection of modulation terms $s(x, \mu) = f^{\text{hyper}}(x, \mu)$, where $s(x, \mu) = [s_1(x, \mu), \ldots, s_L(x, \mu)]$ provides layer-wise adjustments. These modulation terms are injected into each Fourier layer (Eq. (6)) as additive biases, leading to the modified update:

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

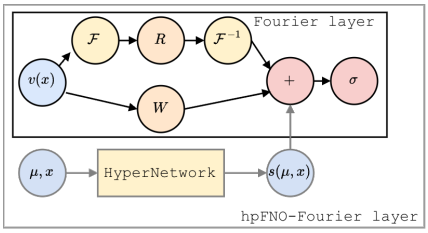


Figure 3: A schematic diagram of $\texttt{hpFNO}_\text{x}$

This mechanism enables the model to adapt its representations according to varying physical parameters while maintaining the underlying operator structure. For more details, we refer readers to Appendix B.1.

4 METHODS

We consider modeling time-evolution of 1D CCP problem, which produces solution trajectories $\{u(x,t;\mu)\}_{t=0}^{T}$ with the initial condition $u(x,0;\mu) = u(0)$. NOs are formulated as one-step evolution operators that approximate the discrete-time dynamics induced by the underlying PDE. In particular, given a finite history of previous solution states $\{u(t-\tau\Delta_t)\}_{\tau=0}^{T_{\text{in}}-1} \in \mathcal{A}$, the operator predicts the subsequent state $u(t+\Delta_t) \in \mathcal{U}$: $G_\Theta : \mathcal{A} \to \mathcal{U}$. Multi-step predictions are then generated autoregressively by repeatedly applying the same operator.

4.1 ENFORCING COUPLED DAE STRUCTURE

Interpreting the Poisson equation as a constraint on the physical fields, the governing equations (Eqs. (1)–(4)) can be viewed as exhibiting a differential–algebraic structure; the evolution equations are coupled with an elliptic Poisson constraint. Since this constraint can be exactly enforced by solving the Poisson equation using a conventional numerical solver, we adopt a solver-in-the-loop (SITL) approach. The forward computational flow can be succinctly represented as follows:

$$(\phi, n_{\text{e}}, n_{\text{i}}, u_{\text{i}})(x,t) \xrightarrow{\texttt{hpFNO}_{\text{x}}} (n_{\text{e}}, n_{\text{i}}, u_{\text{i}})(x, t+\Delta t) \xrightarrow{\text{Poisson solve}} \phi(x, t+\Delta t).$$

That is, $\texttt{hpFNO}_{\text{x}}$ models the evolution operator, while the Poisson solver enforces the electric potential to satisfy the Poisson equation. We refer to this as $\texttt{hpFNO}_{\text{x}}$ with Poisson closure, $\texttt{hpFNO}_{\text{x}}\texttt{-PC}$.

4.2 FLOQUET–POINCARÉ REGULARIZATION FOR RF-PERIODIC DYNAMICS

Recall the solution profiles exhibit two distinct temporal scales (see Figure 2): the electric potential undergoes rapid oscillations within each RF cycle, directly reflecting the applied boundary condition, while the density and velocity profiles evolve more gradually from one cycle to the next. The trajectories therefore combine fast intra-cycle dynamics with a slower cycle-to-cycle relaxation toward a quasi-periodic response.

The $\texttt{hpFNO}_{\text{x}}\texttt{-PC}$ backbone models the local dynamics by repeatedly applying a learned Δt-step evolution operator. This local formulation is necessary for resolving the rapidly oscillating solution profiles, but it does not explicitly use the fact that states separated by one RF period correspond to the same phase of the external forcing. Moreover, a small local prediction error can accumulate through repeated autoregressive composition over many RF cycles. We therefore introduce an additional training objective that constrains the learned dynamics at the physically distinguished time scale T_{RF}. Figure 4 illustrates the difference between the new auxiliary cycle-to-cycle operator and the $\texttt{hpFNO}_{\text{x}}\texttt{-PC}$ backbone model.

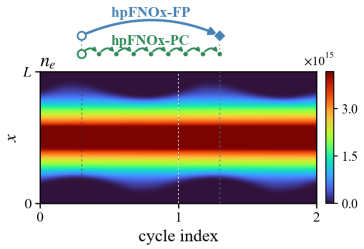


Figure 4: An illustrative description of the local operator $\texttt{hpFNO}_{\text{x}}\texttt{-PC}$ (green) and an auxiliary cycle-to-cycle operator $\texttt{hpFNO}_{\text{x}}\texttt{-FP}$ (blue).

Poincaré map and Floquet motivation. Consider a generic differential variable z evoling under a periodically forced system, $\partial_t z = f(z,t)$ with $f(z,t) = f(z, t+T_{\text{RF}})$. For a fixed forcing phase, the stroboscopic Poincaré map advances the state by exactly one forcing period, $\mathcal{P} : z(t) \mapsto z(t+T_{\text{RF}})$ (Guckenheimer & Holmes, 1983; Kuznetsov, 2004). A T_{RF}-periodic response therefore corresponds to a fixed point of this map, satisfying $\mathcal{P}(z^*) = z^*$. Linearizing the one-period map around z^* yields the monodromy operator, whose eigenvalues, known as Floquet multipliers, characterize whether infinitesimal perturbations are amplified or attenuated over one forcing period (Floquet, 1883; Kuznetsov, 2004). This perspective identifies the one-period return map as a natural object for constraining the stability of long autoregressive rollouts. Although our method does not explicitly compute the monodromy operator or its Floquet multipliers, it uses the phase-aligned one-period evolution underlying Poincaré and Floquet analyses as an auxiliary training signal. We refer to this approach as *Floquet–Poincaré-inspired* (FP) regularization.

Auxiliary one-cycle neural operator. To instantiate the FP regularization, we introduce an auxiliary NO that captures the slower cycle-to-cycle relaxation behavior: given a finite history states $\mathbf{U}_t =$

$\{u(t-\tau\Delta_t\}_{\tau=0}^{T_{\text{in}}-1}$ directly predicts the state in one RF cycle later, $u(t+T_{\text{RF}}) \in \mathcal{U} : \mathcal{P}_{\Theta_{\text{P}}} : \mathcal{A} \to \mathcal{U}$:

$$\widehat{P}_{\Theta_{\text{P}}} : \mathbf{U}_t \mapsto u(t+T_{\text{TF}}).$$

As in $\texttt{hpFNO}_{\text{x}}\texttt{-PC}$, the NO inputs the four-field history, the spatial coordinate, and the physical parameter, and predicts the next state of the three variables, $\widehat{u}^{\text{FP}} = (\widehat{n}_e^{\text{FP}}, \widehat{n}_i^{\text{FP}}, \widehat{u}_i^{\text{FP}})$; then, the electric potential is reconstructed using the same Poisson closure: $(\widehat{n}_e^{\text{FP}}, \widehat{n}_i^{\text{FP}}, \widehat{u}_i^{\text{FP}}) \to \widehat{\phi}^{\text{FP}}$.

This auxiliary model is also trained via supervising against the ground-truth solutions, and its predictions (i.e., next RF states) are leveraged to guide the backbone model prediction as an extra supervising signal. The two training-time computational paths can be summarized as

$$\mathbf{U}_t \xrightarrow[\text{local } \Delta t\text{-scale rollout}]{\texttt{hpFNO}_{\text{x}}\texttt{-PC}} \widehat{\mathbf{u}}^{\text{PC}}(t+\Delta_t), \quad \text{and} \quad \mathbf{U}_t \xrightarrow[\text{one-RF-cycle jump}]{\texttt{hpFNO}_{\text{x}}\texttt{-FP}} \widehat{\mathbf{u}}^{\text{FP}}(t+\Delta_t),$$

where the superscripts $^{\text{PC}}$ and $^{\text{FP}}$ are used to differentiate the predictions made by the local rollout operator and the cycle-to-cycle prediction operator. Then, the discrepancy between $\widehat{\mathbf{u}}^{\text{PC}}$ and $\widehat{\mathbf{u}}^{\text{FP}}$ is minimized during training. During inference, only the backbone local operator is used and we refer this model as $\texttt{hpFNO}_{\text{x}}\texttt{-FP}$.

4.3 Scaling training strategy for long trajectories.

Autoregressive rollout models can be trained either with teacher forcing (TF), where ground-truth states are provided as inputs, or with multi-step autoregressive predictions. For long trajectories, however, training over the entire temporal horizon can be computationally expensive regardless of the training strategy. To enable scalable training on long trajectories, we introduce temporal subsequence sampling into the training of autoregressive NOs. Specifically, instead of processing an entire trajectory at each training iteration, we randomly sample shorter contiguous subsequences and optimize the model over these sampled temporal windows. Although subsequence-based training is well established for sequential dynamical models (Chen et al., 2018), its application to long-horizon NO training has not, to our knowledge, been systematically investigated.

Algorithm 1 illustrates the random temporal subsequence sampling procedure for generic time-series data $\mathbf{U} \in \mathbb{R}^{B\times T\times d}$, where B denotes the number of trajectories, T the trajectory length, and d the state dimension (e.g., the number of species). For each sampled subsequence, a trajectory and a valid temporal starting point are independently selected at random. A contiguous window of length L_{sub} is then extracted from the selected trajectory. Repeating this procedure N_{b} times yields a subsequence batch $\mathbf{U}_{\text{sub}} \in \mathbb{R}^{N_{\text{b}}\times L_{\text{sub}}\times d}$. This sampling strategy allows the model to encounter different temporal regions and initial states across training iterations without processing the entire trajectory at each iteration, thereby reducing the computational cost of training on long trajectories.

Algorithm 1 Random Temporal Subsequence Sampling

Input: Trajectory data $\mathbf{U} \in \mathbb{R}^{B\times T\times d}$, number of subsequences N_{b}, subsequence length L_{sub}
Output: Subsequence batch $\mathbf{U}_{\text{sub}} \in \mathbb{R}^{N_{\text{b}}\times L_{\text{sub}}\times d}$

1: Sample trajectory indices: $\mathbf{b} = (b_1, \ldots, b_{N_{\text{b}}})$, where $b_i \overset{\text{i.i.d.}}{\sim} \text{Uniform}\{1, \ldots, B\}$
2: Sample temporal indices: $\mathbf{s} = (s_1, \ldots, s_{N_{\text{b}}})$, where $s_i \overset{\text{i.i.d.}}{\sim} \text{Uniform}\{1, \ldots, T - L_{\text{sub}} + 1\}$
3: Form the subsequence batch: $(\mathbf{U}_{\text{sub}})_{i,k} = \mathbf{U}_{b_i,\, s_i+k-1}, \quad (i,k) \in [N_{\text{b}}] \times [L_{\text{sub}}]$

4.4 Putting all together

At every training iteration, a set of subsequences is sampled via the algorithm described in Section 4.3 and used as supervising signal for training NOs. The standard $\texttt{hpFNO}_{\text{x}}\texttt{-PC}$ loss supervises an autoregressive subsequence of length, L_{sub}: $\mathcal{L}_{\Delta_t} = \sum_{k=0}^{L_{\text{sub}}-1} \ell(\widehat{\mathbf{u}}^{\text{PC}}(t+k\Delta_t), \mathbf{u}(t+k\Delta_t))$; the auxiliary $\texttt{hpFNO}_{\text{x}}\texttt{-FP}$ loss supervises a cycle-to-cycle prediction of length L_{FP}: $\mathcal{L}_{\text{P}} = \sum_{k=0}^{L_{\text{FP}}-1} \ell(\widehat{\mathbf{u}}^{\text{FP}}(t + T_{\text{RF}} + k\Delta_t), \mathbf{u}(t + T_{\text{RF}} + k\Delta_t))$ and, lastly, the FP regularization is enforced by minimizing $\mathcal{L}_{\text{FP}} = \ell(\widehat{\mathbf{u}}^{\text{PC}}(t+T_{\text{RF}}), \widehat{\mathbf{u}}^{\text{FP}}(t+T_{\text{RF}}))$. The complete objective becomes $\min_{\Theta,\Theta_{\text{P}}} \mathcal{L}_{\Delta t}(\Theta) + \lambda_{\text{P}}\mathcal{L}_{\text{P}}(\Theta_{\text{P}}) + \lambda_{\text{FP}}\mathcal{L}_{\text{FP}}(\Theta_{\text{P}})$, where λ_{P} and λ_{FP} denote weighting scalars. Full details on the training algorithm can be found in Appendix B.3.

5 EXPERIMENTAL RESULTS

This section presents results of experimentation comparing the proposed methods with various baseline models. All models are implemented in PYTHON with PYTORCH (Paszke et al., 2019). The models in the FNO model family are built upon the original implementation of FNOs (Li et al., 2021) and the implementation of the coupled and parameterized extension (i.e., $\texttt{hpFNO}_\text{x}$) follows Jing et al. (2026). Each experiment is repeated four times with different random seeds, and all computations are performed on an NVIDIA RTX 5090. The details are in Appendix B.

5.1 EXPERIMENTAL SETUP

Task objective As described in the method section, the operator predicts the subsequent state $u(t+\Delta_t)$ given a finite history of previous solution states $\mathbf{u}_t = \{u(t-\tau\Delta_t)\}_{\tau=0}^{T_\text{in}-1}$, $u(t+\Delta_t) = G_\theta(\mathbf{u}_t;\mu)$. Multi-step predictions are then generated autoregressively by repeatedly applying the same operator, $u(t+n\Delta_t) = G^{\circ n}(\mathbf{u}_t;\mu)$. Both training and testing are initialized with the ground-truth input window $(u(T_\text{in}-1),\ldots,u(0))$. Subsequent rollouts update the input window with predicted states during non-TF training and testing, and with ground-truth states during TF training.

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: $\text{err} = \sum_{i\in\{\phi,n_\text{e},n_\text{i},u_\text{i}\}} \text{nRMSE}^i$. For all methods, reported errors are computed on the original physical scale.

5.2 DATA GENERATION

Training data are generated by solving the governing equations numerically with a finite-difference scheme. We use 128 spatial grid cells and resolve each forcing cycle with 256 temporal steps, where the cycle period corresponds to the 13.56 MHz boundary excitation frequency. All simulations are initialized from the same initial state, allowing the learning task to emphasize the evolution of the underlying physical dynamics rather than variations in initial conditions. The resulting trajectories are subsequently downsampled by a factor of two in time, producing 128 snapshots per trajectory.

A physical parameter is sampled on an equidistant grid, which contains 100 elements, $R_0 \in [2.7\times 10^{18}, 2.7\times 10^{21}]$, which are then split into 80/10/10 for training/validation/test samples. Please find more details in Appendix A.10.

5.3 RESULTS

Main performance comparison First we present the findings of the main experiments, the long rollout experiments. We consider trajectories consists of 48 RF cycles (temporally downsampled to have 6144 inference steps). We present the performance of the proposed $\texttt{hpFNO}_\text{x}\texttt{-PC}$ (Poisson closure) and $\texttt{hpFNO}_\text{x}\texttt{-FP}$ (Floquet–Poincarè regularized). For comparison baselines, we consider $\texttt{hpFNO}_\text{x}$ (Jing et al., 2026) and its BC-enforced version, $\texttt{hpFNO}_\text{x}\texttt{-BC}$, which is inspired from recent development NOs for enforcing BC (Wang et al., 2024; Mousavi et al., 2026). Additionally, we consider several representative NOs: FNO (Li et al., 2021), FFNO (Tran et al., 2023), CNO (Raonic et al., 2023), S4FFNO (Buitrago Ruiz et al., 2025), and DISCO (Morel et al., 2025).

Table 1 presents the performance measured in nRMSE for each physical variable (ϕ, n_e, n_i, u_i) and the overall performance (i.e., measured as the sum of nRMSEs of each physical variable). $\texttt{hpFNO}_\text{x}\texttt{-FP}$ achieves the best performance ($\sim$6% relative error overall) reducing the prediction error by 79.8% relative to $\texttt{hpFNO}_\text{x}$. It is noticeable that introducing the BC enforcement (embedded in all three $\texttt{hpFNO}_\text{x}$ variants) dramatically reduces the learning difficulty caused by oscillatory behavior in electric potential ϕ due to the RF-forcing. While achieving improvement prediction capability for ϕ, $\texttt{hpFNO}_\text{x}\texttt{-BC}$ struggles model the other three variables, resulting in slight overall improvement compared with $\texttt{hpFNO}_\text{x}$.

Table 1: Performance comparison (values are scaled by 10^{-1} and formatted as mean ($\pm$ standard deviation)): A long rollout experiment with 48RF cycle trajectories (6144 temporal steps). The individual prediction errors are measured for each physical variable (ϕ, n_e, n_i, u_i) and the overall performance (i.e., the sum of all nRMSEs).

Method	Overall	ϕ	n_e	n_i	u_i
FNO	12.880 ($\pm$ 2.4248)	2.3953 ($\pm$ 2.1401)	4.9611 ($\pm$ 0.1970)	4.7615 ($\pm$ 0.2012)	0.7617 ($\pm$ 0.1609)
FFNO	14.773 ($\pm$ 3.3882)	4.1104 ($\pm$ 2.3256)	4.8710 ($\pm$ 0.3366)	4.6459 ($\pm$ 0.3181)	1.1461 ($\pm$ 0.4838)
CNO	16.066 ($\pm$ 2.5815)	4.1942 ($\pm$ 2.2148)	5.3709 ($\pm$ 0.3985)	5.1459 ($\pm$ 0.4369)	1.3547 ($\pm$ 0.7118)
S4FFNO	9.2859 ($\pm$ 0.2149)	0.7245 ($\pm$ 0.5455)	4.0993 ($\pm$ 0.3123)	3.9173 ($\pm$ 0.2904)	0.5447 ($\pm$ 0.2266)
DISCO	26.316 ($\pm$ 6.3248)	6.4827 ($\pm$ 1.1690)	7.8321 ($\pm$ 1.5851)	6.2441 ($\pm$ 1.3574)	5.7568 ($\pm$ 2.9130)
hpFNO$_x$	2.8836 ($\pm$ 2.9906)	1.8984 ($\pm$ 2.3278)	0.4193 ($\pm$ 0.3376)	0.3191 ($\pm$ 0.1670)	0.2468 ($\pm$ 0.1681)
hpFNO$_x$-BC	2.6763 ($\pm$ 3.1504)	0.2288 ($\pm$ 0.3070)	0.9847 ($\pm$ 1.1186)	0.9416 ($\pm$ 1.0567)	0.5212 ($\pm$ 0.6824)
hpFNO$_x$-PC	1.8419 ($\pm$ 2.6927)	0.1778 ($\pm$ 0.1588)	0.6318 ($\pm$ 0.9759)	0.6176 ($\pm$ 0.9459)	0.4146 ($\pm$ 0.6152)
hpFNO$_x$-FP	0.5828 ($\pm$ 0.2421)	0.1113 ($\pm$ 0.0217)	0.1806 ($\pm$ 0.0879)	0.1715 ($\pm$ 0.0847)	0.1194 ($\pm$ 0.0595)

Cycle-to-cycle convergence Next, Figure 5 illustrates cycle-to-cycle convergence, an important problem-specific characteristic of CCP dynamics. Recall the solution profile (Figure 2); the applied RF voltage periodically modulates the electric field and electrode sheaths, producing strongly oscillatory electron and potential dynamics within each RF cycle. In contrast, the ion dynamics and the global balance among plasma generation, transport, and wall losses evolve over much longer time scales, causing the effects of the initial condition to persist over multiple cycles. As these slower processes equilibrate, the discharge approaches a *periodic quasi-steady state*, or limit cycle, in which spatial solution profiles sampled at the same RF phase increasingly repeat from one cycle to the next. The cycle-convergence metric quantifies this relaxation through the normalized difference between such phase-aligned profiles in two consecutive RF cycles (See Appendix for the definition). Identifying this regime is important because physically meaningful discharge characteristics should be evaluated only after the initialization transient has sufficiently decayed. Figure 5 indicates that the FP-regularized NO, hpFNO$_x$-FP closely follows the cycle-to-cycle convergence behavior of the ground truth trajectory (black dashed line). While hpFNO$_x$-PC produces lower error among the hpFNO$_x$ variants, the cycle-to-cycle convergence results provide a new angle that its convergence behavior is less ideal.

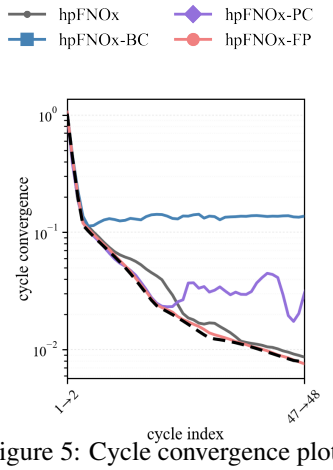


Figure 5: Cycle convergence plot

Performance comparisons for varying trajectory lengths Next, we vary the length of trajectories by varying the number of RF cycles: $N_{\mathrm{RF}} = \{2, 4, 8, 16, 24, 32, 40, 48\}$ and test all four variants of hpFNO$_x$. For each value of N_{RF}, the training protocol across the compared methods remain the same. Figure 6 reports the results. When N_{RF} is small, all four variants perform similarly; the vanilla hpFNO$_x$ indeed achieves the smallest error (the margin relative to hpFNO$_x$-PC is very small though); hpFNO$_x$-PC demonstrates the best performance for short-to-moderate length trajectories (from 2–24). As the trajectory length becomes longer ($\geq$32), the benefits of using the FP-regularization are more pronounced, achieving the least increment in the prediction errors for longer trajectories. The shaded areas present the standard deviation of the errors; hpFNO$_x$-FP demonstrates robustness towards random seed sensitivity while all three other variants exhibit sensitivity towards random initialization.

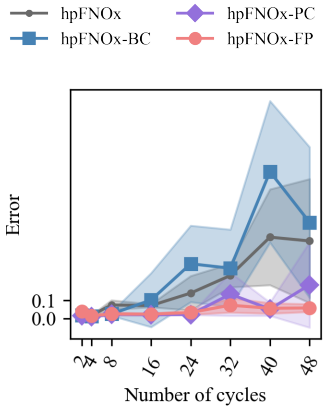


Figure 6: Performance comparisons for varying trajectory lengths.

Performance comparisons for varying training parameters Under random temporal subsequence training (Section 4.3), N_{in} denotes the number of optimizer updates performed per epoch, each using a newly sampled batch of temporal windows (see Appendix B.3). We evaluate the model performance on varying subsequence sampling-related hyper-parameters, the number inner iterations N_{in} and the length of sampled subsequence L_{sub}. Along with those, we further compare the results obtained via using existing training schemes, namely, full auto-regressive rollout (AR), full teacher forcing (TF), and scheduled TF, which starts with full TF at the beginning of the training epoch, but ends with full AR at the end of training. Since training schemes without subsequence sampling causes memory issues with very long trajectories, we conduct the experiments with trajectories with four RF cycles (512 time steps). The full-horizon training schemes require substantially longer wall-clock time per epoch without improving accuracy compared to the random sampling method. In the random sampling method, while it overall achieves lower errors with reduced timing, non-monotonic trade-off between training cost and predictive accuracy exists, requiring proper hyper-parameter configuration for the optimized performance.

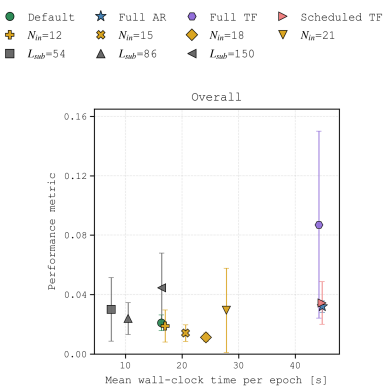


Figure 7: Performance comparisons for varying hyper-parameters and training schemes

6 Related Work

Neural-numerical hybrid solvers. Hybrid approaches leverage established numerical components while learning corrections, or unresolved physical operators. Representative examples include differentiable solver-in-the-loop correction Um et al. (2020), indirect neural correction, which embeds learned correction terms within the governing equations rather than modifying solver outputs Wei et al. (2025), HINTS, which combines DeepONet Lu et al. (2021) predictions with classical relaxation iterations Zhang et al. (2024), and operator-splitting approaches learning nonlinear physical operators while retaining fixed numerical approximations for known linear terms Gopakumar et al. (2026). In contrast, our NO serves as the primary time-advancement model for the subsets of physical variables, while the numerical component is restricted to reconstructing the algebraic electric potential.

Constrained and differential–algebraic learned dynamics. Known algebraic constraints have been incorporated into neural differential equations. Representative examples include stabilization toward constraint manifolds White et al. (2023), operator-split neural DAEs Koch et al. (2025), and post-step projection of predicted states Pal et al. (2025). Our approach exploits a specific semi-explicit structure: the Poisson equation determines the electric potential from the charged-species densities and boundary data. We therefore eliminate the algebraic variable through a numerical Poisson solve.

Long-horizon autoregressive learning. Autoregressive surrogate models are susceptible to distribution shift and compounding prediction errors over long rollouts. Prior approaches address these issues through the "pushforward" trick Brandstetter et al. (2022), iterative prediction refinement Lippe et al. (2023), and explicit temporal memory Buitrago Ruiz et al. (2025). Our work considers a complementary scalability problem arising from very long training trajectories. We proposed to train on randomly sampled contiguous subsequences.

7 Conclusion

We developed a CCP-tailored neural operator for long-horizon prediction of RF-driven plasma dynamics. At every autoregressive step, Poisson closure enforces the elliptic constraint and substantially reduces electric-potential error, while Floquet–Poincaré-inspired regularization uses phase-aligned one-cycle supervision to reduce drift across RF cycles. Random temporal subsequence training enables scalable optimization without processing full trajectories. On the parameterized 1D CCP benchmark, $\texttt{hpFNO}_{\text{x}}\texttt{-FP}$ achieves approximately 6% overall error, a 79.8% reduction relative to $\texttt{hpFNO}_{\text{x}}$, while more accurately capturing cycle-to-cycle convergence. These results demonstrate the value of incorporating problem-specific algebraic and periodic structures into neural operators for long-horizon plasma simulation.

AI USE STATEMENT

In this work, we used generative AI tools, including large language models (LLMs), for language editing, literature discovery, coding assistance, and clarification of submission guidelines. For manuscript preparation, LLMs were used to identify grammatical errors, improve clarity and readability, and check the consistency of mathematical notation. All AI-assisted text was reviewed and revised by the authors to ensure technical accuracy and consistency with the intended scientific claims.

LLMs were also used to identify potentially relevant references. The suggested references were manually examined by the authors for relevance and accuracy, and no reference was included solely on the basis of an LLM response. For references retained in the manuscript, bibliographic metadata was obtained or verified through sources such as Google Scholar, DBLP, the publisher's website, or the original paper.

For software development, LLMs assisted with implementing or adapting model components for selected baseline methods while preserving the main training and evaluation pipeline, which was developed by the authors. When available, the corresponding author-provided GitHub repositories were supplied to the LLM as implementation references. LLMs also assisted with Python scripts for visualization and the preparation of schematic diagrams. All AI-assisted code was manually inspected, executed, tested, and modified by the authors before being used to produce the reported results.

Finally, we consulted LLMs for preliminary clarification of author guidelines and submission requirements. Any relevant guidance was subsequently checked against the official conference policies. The authors reviewed all AI-assisted work and take full responsibility for the final manuscript, scientific claims, code, figures, and other artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work concerns numerical modeling and machine-learning surrogates for a simulated plasma physics system. It does not involve human participants, personal or sensitive data, or automated decision-making affecting individuals or social groups. We therefore did not identify any significant ethical concerns specific to this study beyond the standard requirements of scientific rigor, transparency, and responsible reporting.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the governing equations, numerical data-generation procedure, discretization, and parameter sweep in Appendix A. The model architectures, training algorithm, optimization settings, hyperparameters, data partitioning, and model-selection protocol are documented in Appendix B and the experimental setup. Upon acceptance, we will release the complete source code, including environment specifications, data-generation and preprocessing scripts, model implementations, training and evaluation configurations, random seeds, and visualization scripts. The repository will also provide instructions for regenerating the simulation data and reproducing the reported tables and figures, and will serve as an executable specification of low-level implementation details that are impractical to enumerate in the manuscript.

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.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

Johannes Brandstetter, Daniel E. Worrall, and Max Welling. Message passing neural PDE solvers. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=vSix3HPYKSU.

Ricardo Buitrago Ruiz, Tanya Marwah, Albert Gu, and Andrej Risteski. On the benefits of memory for modeling time-dependent pdes. In *International Conference on Learning Representations*, volume 2025, pp. 54972–55002, 2025.

Naomi Carey, Lorenzo Zanisi, Stanislas Pamela, Vignesh Gopakumar, John Omotani, James Buchanan, Johannes Brandstetter, Fabian Paischer, Gianluca Galletti, and Paul Setinek. Neural operator surrogate models of plasma edge simulations: feasibility and data efficiency. *Nuclear Fusion*, 65(10):106010, 2025.

Pascal Chabert and Nicholas Braithwaite. *Physics of Radio-Frequency Plasmas*. Cambridge University Press, 2011.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Gaston Floquet. Sur les équations différentielles linéaires à coefficients périodiques. In *Annales scientifiques de l'École normale supérieure*, volume 12, pp. 47–88, 1883.

Vignesh Gopakumar, Stanislas Pamela, Lorenzo Zanisi, Zongyi Li, Ander Gray, Daniel Brennand, Nitesh Bhatia, Gregory Stathopoulos, Matt Kusner, Marc Peter Deisenroth, et al. Plasma surrogate modelling using Fourier neural operators. *Nuclear Fusion*, 64(5):056025, 2024.

Vignesh Gopakumar, Ander Gray, Daniel Giles, Lorenzo Zanisi, Matt J. Kusner, Timo Betcke, Stanislas Pamela, and Marc Peter Deisenroth. Learning physical operators using neural operators. In *Proceedings of The 29th International Conference on Artificial Intelligence and Statistics*, volume 300, pp. 3223–3231. PMLR, 2026.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in neural information processing systems*, 35:35971–35983, 2022.

John Guckenheimer and Philip Holmes. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, volume 42 of *Applied Mathematical Sciences*. Springer, New York, 1983. doi: 10.1007/978-1-4612-1140-2.

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

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Cheng Jing, Uvini Balasuriya Mudiyanselage, Abhishek Verma, Kallol Bera, Shahid Rauf, and Kookjin Lee. Extending Fourier neural operators for modeling parameterized and coupled PDEs. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=rtUT5Wic10.

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.

James Koch, Madelyn Shapiro, Himanshu Sharma, Draguna Vrabie, and Ján Drgoňa. Learning neural differential algebraic equations via operator splitting. In *2025 IEEE 64th Conference on Decision and Control (CDC)*, pp. 2077–2084. IEEE, 2025.

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.

Yuri A. Kuznetsov. *Elements of Applied Bifurcation Theory*. Springer, New York, 3 edition, 2004. doi: 10.1007/978-1-4757-3978-7.

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.

Michael A. Lieberman and Allan J. Lichtenberg. *Principles of Plasma Discharges and Materials Processing*. Wiley, New York, 1994. ISBN 978-0-471-00577-3.

Phillip Lippe, Bas Veeling, Paris Perdikaris, Richard Turner, and Johannes Brandstetter. PDE-refiner: Achieving accurate long rollouts with neural PDE solvers. *Advances in Neural Information Processing Systems*, 36:67398–67433, 2023.

EM Liston, Ludvik Martinu, and Michael R Wertheimer. Plasma surface modification of polymers for improved adhesion: a critical review. *Journal of adhesion science and technology*, 7(10): 1091–1127, 1993.

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.

Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, pp. 3. Atlanta, GA, 2013.

Rudy Morel, Jiequn Han, and Edouard Oyallon. DISCO: learning to DISCover an evolution operator for multi-physics-agnostic prediction. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=6EZ3MDDf6p.

Sepehr Mousavi, Siddhartha Mishra, and Laura De Lorenzis. Imposing boundary conditions on neural operators via learned function extensions. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=gnzNB7GFFo.

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.

Avik Pal, Alan Edelman, and Christopher Rackauckas. Semi-explicit neural daes: Learning long-horizon dynamical systems with algebraic constraints. *arXiv preprint arXiv:2505.20515*, 2025.

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, 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.

Bogdan Raonic, Roberto Molinaro, Tim De Ryck, Tobias Rohner, Francesca Bartolucci, Rima Alaifari, Siddhartha Mishra, and Emmanuel De Bézenac. Convolutional neural operators for robust and accurate learning of PDEs. *Advances in Neural Information Processing Systems*, 36: 77187–77200, 2023.

Shahid Rauf, Manuel Schroeder, Ihor Korolov, Jason Kenney, and Julian Schulze. Plasma dynamics in a capacitively coupled discharge driven by a combination of a single high frequency and a tailored low frequency rectangular voltage waveform. *Plasma Sources Science and Technology*, 32 (3):034002, 2023.

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

Kiwon Um, Robert Brand, Yun Raymond Fei, Philipp Holl, and Nils Thuerey. Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers. *Advances in neural information processing systems*, 33:6111–6122, 2020.

Haixin Wang, Jiaxin Li, Anubhav Dwivedi, Kentaro Hara, and Tailin Wu. BENO: Boundary-embedded neural operators for elliptic PDEs. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 8308–8329, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/218ca0d92e6ed8f9db00621e103dc70c-Paper-Conference.pdf.

Hao Wei, Aleksandra Franz, Bjoern List, and Nils Thuerey. INC: An indirect neural corrector for auto-regressive hybrid PDE solvers. *Advances in Neural Information Processing Systems*, 38: 110182–110216, 2025.

Alistair White, Niki Kilbertus, Maximilian Gelbrecht, and Niklas Boers. Stabilized neural differential equations for learning dynamics with explicit constraints. *Advances in Neural Information Processing Systems*, 36:12929–12950, 2023.

Enrui Zhang, Adar Kahana, Alena Kopaničáková, Eli Turkel, Rishikesh Ranade, Jay Pathak, and George Em Karniadakis. Blending neural operators and relaxation methods in PDE numerical solvers. *Nature Machine Intelligence*, 6(11):1303–1313, 2024.

A Reference Plasma Solver and Parameter-Sweep Generation

This section provides a detailed description of the finite-difference solver used to generate the reference trajectories. The implemented residual equations are consistent with a one-dimensional electrostatic drift–diffusion and cold-ion fluid model for the electrostatic potential ϕ, electron density n_e, ion density n_i, and ion velocity u_i. Each trajectory is generated by solving these four fields simultaneously at every time step.

A.1 Governing equations

We consider a one-dimensional spatial domain $x \in [0, L]$ with $L = 2.5 \times 10^{-2}$ m (See Figure 1).

The charge density is determined by singly charged ions and electrons, and Poisson's equation is written as

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{e}{\epsilon_0}(n_i - n_e) = 0,$$

where e is the elementary charge and ϵ_0 is the vacuum permittivity. The electron continuity equation is represented in conservative form,

$$\frac{\partial n_e}{\partial t} + \frac{\partial \Gamma_e}{\partial x} = R(x), \tag{7}$$

with the electron flux

$$\Gamma_e = \mu_e n_e \frac{\partial \phi}{\partial x} - D_e \frac{\partial n_e}{\partial x}.$$

Equivalently,

$$\frac{\partial n_e}{\partial t} = D_e \frac{\partial^2 n_e}{\partial x^2} - \frac{\partial}{\partial x}\left(\mu_e n_e \frac{\partial \phi}{\partial x}\right) + R(x).$$

The electron mobility and diffusion coefficient are defined by

$$\mu_e = \frac{e}{m_e \nu_m}, \qquad D_e = \mu_e T_e,$$

where m_e is the electron mass, ν_m is the prescribed electron momentum-transfer frequency, and $T_e = 3.0$ is the temperature-like coefficient used in the Einstein relation. With $\nu_m = 10^7\ \mathrm{s}^{-1}$, the coefficients used by the discretization correspond to

$$\mu_e \approx 1.75882 \times 10^4\ \mathrm{m^2\,V^{-1}\,s^{-1}}, \qquad \text{and} \qquad D_e \approx 5.27646 \times 10^4\ \mathrm{m^2\,s^{-1}}.$$

The ion density satisfies

$$\frac{\partial n_i}{\partial t} + \frac{\partial}{\partial x}(n_i u_i) = R(x), \tag{8}$$

and the ion velocity is governed by the cold-ion momentum equation

$$\frac{\partial u_i}{\partial t} + \frac{\partial}{\partial x}\left(\frac{u_i^2}{2}\right) + \frac{e}{m_i}\frac{\partial \phi}{\partial x} + \nu_i u_i = 0.$$

The implementation does not include an ion-pressure term. The ion mass and collision frequency are fixed at

$$m_i = 40\, m_{\mathrm{amu}} = 6.64215492 \times 10^{-26}\ \mathrm{kg}, \qquad \text{and} \qquad \nu_i = 10^6\ \mathrm{s}^{-1}.$$

The same volumetric source is added to the electron and ion continuity equations:

$$R(x) = R_0\, \mathbb{I}_{[x_{s,1}, x_{s,2}]}(x), \qquad \text{with} \qquad x_{s,1} = 0.010\ \mathrm{m}, \qquad x_{s,2} = 0.015\ \mathrm{m},$$

where $\mathbb{I}$ denotes the indicator function. The equal source terms in Eqs. (7) and (8) introduce electron–ion pairs without directly injecting net charge.

A.2 BOUNDARY AND INITIAL CONDITIONS

The left electrode is grounded, and the right electrode is driven by a sinusoidal RF voltage:

$$\phi(0,t) = 0, \qquad \phi(L,t) = V_0 \sin(2\pi f_{\mathrm{RF}} t),$$

where $V_0 = 100$ V and $f_{\mathrm{RF}} = 13.56 \times 10^6$ Hz. The corresponding RF period is $T_{\mathrm{RF}} = \frac{1}{f_{\mathrm{RF}}} \approx 7.3746313 \times 10^{-8}$ s. Homogeneous Dirichlet conditions are imposed on the electron density, $n_e(0,t) = n_e(L,t) = 0$. For the ion density and ion velocity, the code applies linear extrapolation at both boundaries. In discrete form, these conditions are

$$\begin{aligned} n_{i,0} - 2n_{i,1} + n_{i,2} &= 0, & n_{i,N_x} - 2n_{i,N_x-1} + n_{i,N_x-2} &= 0, \\ u_{i,0} - 2u_{i,1} + u_{i,2} &= 0, & u_{i,N_x} - 2u_{i,N_x-1} + u_{i,N_x-2} &= 0. \end{aligned}$$

These equations enforce a zero second difference, or equivalently a linear extrapolation of the boundary value from the nearest two interior values. They should not be interpreted as direct zero-gradient or zero-flux boundary conditions.

The initial conditions are $\phi(x,0) = 0, u_i(x,0) = 0, n_i(x,0) = n_{\mathrm{init}}$, where $n_{\mathrm{init}} = 5 \times 10^{14}$ m^{-3}. The electron density is initialized to n_{init} at the interior grid points and is set to zero at the two electrode locations:

$$n_e(x_j,0) = \begin{cases} 0, & j = 0 \text{ or } j = N_x, \\ n_{\mathrm{init}}, & 1 \le j \le N_x - 1. \end{cases}$$

A.3 DETAILED DESCRIPTION ON THE CCP MODEL

We consider a one-dimensional electrostatic fluid model of a symmetric parallel-plate CCP in argon with a 2.5 cm gap. The left electrode is grounded and the right electrode is driven directly by $V(t) = V_0 \sin(2\pi f_{\mathrm{RF}} t)$, with $V_0 = 100V$ and $f_{\mathrm{RF}} = 13.56$ MHz (Figure 1); no external circuit or blocking capacitor is modeled. Electrons obey a drift diffusion equation with a constant temperature, and ions obey continuity and momentum equations with a constant ion–neutral collision frequency. The potential is obtained from Poisson's equation at each instant. Within these assumptions, the model captures sheath formation, RF modulation of the sheaths, and the slow evolution of the plasma density toward an RF-periodic steady state.

All trajectories start from the same initial state: a spatially uniform, quasi-neutral plasma in the interior. This state contains no sheaths or ion flow, does not correspond to any stage of a real discharge (which would evolve from breakdown or from a previous operating point). It satisfies Poisson's equation and is therefore a consistent initial condition for the differential–algebraic system. However, the first several RF cycles of each trajectory are dominated by an artificial start-up transient: electrons are expelled from the electrode regions, and the resulting ion-matrix sheaths relax toward RF sheaths on the ion response time ($\sim 1\mu s$, roughly 10–20 RF cycles). The learning task is therefore to reproduce the transient from a fixed, prescribed initial state, not the evolution of a physically initiated discharge.

A.4 NONDIMENSIONALIZATION

The nonlinear equations are solved in nondimensional form. We define

$$x = x_\star \widehat{x}, \qquad t = t_\star \widehat{t}, \qquad \phi = \phi_\star \widehat{\phi}, \qquad n_s = n_\star \widehat{n}_s, \qquad \text{and} \qquad u_{\mathrm{i}} = u_\star \widehat{u}_{\mathrm{i}},$$

where $s \in \{\mathrm{e}, \mathrm{i}\}$ and $x_\star = 10^{-4}$ m, $t_\star = 10^{-8}$ s, $\phi_\star = 100$ V, $n_\star = 10^{16}$ m^{-3}, with $u_\star = \frac{x_\star}{t_\star} = 10^4$ m s^{-1} and $S_\star = \frac{n_\star}{t_\star} = 10^{24}$ m^{-3} s^{-1}.

The resulting dimensionless domain and fixed parameters are $\widehat{L} = 250, \widehat{x}_{s,1} = 100, \widehat{x}_{s,2} = 150, \widehat{V}_0 = 1, \widehat{f}_{\mathrm{RF}} = f_{\mathrm{RF}} t_\star = 0.1356$, and $\widehat{\omega}_{\mathrm{RF}} = 2\pi \widehat{f}_{\mathrm{RF}} \approx 0.85199993$. The normalized collision frequencies and initial density are $\widehat{\nu}_m = \nu_m t_\star = 0.1, \widehat{\nu}_i = \nu_i t_\star = 0.01$, and $\widehat{n}_{\mathrm{init}} = \frac{n_{\mathrm{init}}}{n_\star} = 0.05$.

The dimensionless governing equations can be expressed as

$$\begin{aligned}
\frac{\partial \widehat{n}_e}{\partial \widehat{t}} - \mathcal{D}_e \frac{\partial^2 \widehat{n}_e}{\partial \widehat{x}^2} + \mathcal{M}_e \frac{\partial}{\partial \widehat{x}} \left(\widehat{n}_e \frac{\partial \widehat{\phi}}{\partial \widehat{x}} \right) &= \widehat{R}, \\
\frac{\partial \widehat{n}_i}{\partial \widehat{t}} + \frac{\partial}{\partial \widehat{x}} \left(\widehat{n}_i \widehat{u}_i \right) &= \widehat{R}, \\
\frac{\partial \widehat{u}_i}{\partial \widehat{t}} + \frac{\partial}{\partial \widehat{x}} \left(\frac{\widehat{u}_i^2}{2} \right) &= -\mathcal{A}_i \frac{\partial \widehat{\phi}}{\partial \widehat{x}} - \widehat{\nu}_i \widehat{u}_i, \\
\frac{\partial^2 \widehat{\phi}}{\partial \widehat{x}^2} &= -\lambda_\phi \left(\widehat{n}_i - \widehat{n}_e \right),
\end{aligned}$$

where

$$\lambda_\phi = \frac{e n_\star x_\star^2}{\epsilon_0 \phi_\star} \approx 1.8095101 \times 10^{-2}, \qquad \mathcal{D}_e = \frac{D_e t_\star}{x_\star^2} \approx 5.2764605 \times 10^4,$$
$$\mathcal{M}_e = \frac{\mu_e \phi_\star t_\star}{x_\star^2} \approx 1.7588202 \times 10^6, \qquad \mathcal{A}_i = \frac{e \phi_\star t_\star^2}{m_i x_\star^2} \approx 2.4121335.$$

The normalized source is $\widehat{R} = \widetilde{R}\, \mathbb{I}_{[100,150]}(\widehat{x})$, where $\widetilde{R} = \frac{R}{S_\star}$. For the parameter range used in the sweep, $\widetilde{R} \in \left[2.7 \times 10^{-6}, 2.7 \times 10^{-3}\right]$.

A.5 Spatial and temporal grids

The dimensionless domain is divided into $N_x = 127$ uniform spatial intervals, resulting in $N_p = N_x + 1 = 128$ grid points. Each RF period is discretized using $N_t = 256$ fixed time steps: $\Delta t = \frac{T_{\mathrm{RF}}}{N_t} = \frac{1}{f_{\mathrm{RF}} N_t} \approx 2.88071534 \times 10^{-10}$ s. Each trajectory contains 48 complete RF cycles and therefore

$$N_{\mathrm{snapshot}} = 48 \times 256 = 12{,}288$$

time steps. The corresponding simulated physical duration is

$$48 T_{\mathrm{RF}} \approx 3.5398230 \times 10^{-6}\ \mathrm{s}.$$

A.6 Fully implicit finite-difference discretization

For compactness, we remove hats in notations, but all variables should be understood as nondimensional. Let $\mathbf{u}^{n+1} = \left[\boldsymbol{\phi}^{n+1}, \mathbf{n}_e^{n+1}, \mathbf{n}_i^{n+1}, \mathbf{u}_i^{n+1}\right] \in \mathbb{R}^{4N_p}$. The dynamic equations are advanced with backward Euler. All diffusive, drift, advective, electrostatic-force, and collision terms are evaluated using the new-time unknowns. Poisson's equation is enforced algebraically at the same nonlinear solve.

Poisson equation. For $j = 1, \ldots, N_x - 1$, the Poisson residual is

$$\mathcal{R}_{\phi,j} = \phi_{j+1}^{n+1} - 2\phi_j^{n+1} + \phi_{j-1}^{n+1} + C_\phi \left(n_{i,j}^{n+1} - n_{e,j}^{n+1} \right) = 0.$$

This is the standard three-point second-order centered approximation of the second derivative.

Electron continuity equation. For an interior node j, the electron residual is

$$\begin{aligned}
\mathcal{R}_{e,j} = n_{e,j}^{n+1} - n_{e,j}^{n} - \Delta t \Bigg[& C_D \left(n_{e,j-1}^{n+1} - 2n_{e,j}^{n+1} + n_{e,j+1}^{n+1} \right) \\
& - C_\mu \Big[\left(n_{e,j+1}^{n+1} + n_{e,j}^{n+1} \right) \left(\phi_{j+1}^{n+1} - \phi_j^{n+1} \right) - \left(n_{e,j-1}^{n+1} + n_{e,j}^{n+1} \right) \left(\phi_j^{n+1} - \phi_{j-1}^{n+1} \right) \Big] + S_j \Bigg] = 0,
\end{aligned}$$

where $C_D = \frac{\mathcal{D}_e}{\Delta x^2} \approx 1.3616645 \times 10^4$, and $C_\mu = \frac{\mathcal{M}_e}{2\Delta x^2} \approx 2.2694408 \times 10^5$.

Equivalently, the electron scheme can be written in conservative flux form as

$$\frac{n_{e,j}^{n+1} - n_{e,j}^{n}}{\Delta t} + \frac{\Gamma_{e,j+1/2}^{n+1} - \Gamma_{e,j-1/2}^{n+1}}{\Delta x} = S_j,$$

with

$$\Gamma_{e,j+1/2}^{n+1} = \mathcal{M}_e \left(\frac{n_{e,j}^{n+1} + n_{e,j+1}^{n+1}}{2} \right) \frac{\phi_{j+1}^{n+1} - \phi_j^{n+1}}{\Delta x} - \mathcal{D}_e \frac{n_{e,j+1}^{n+1} - n_{e,j}^{n+1}}{\Delta x}.$$

Thus, both the diffusive term and the centered drift flux are formally second-order accurate in space for sufficiently smooth solutions.

Ion continuity equation. The ion mass flux is evaluated with a first-order upwind flux splitting. Define

$$u_j^+ = \frac{u_j + |u_j|}{2}, \qquad u_j^- = \frac{u_j - |u_j|}{2}.$$

The numerical flux associated with the interface between density values $n_{i,j}$ and $n_{i,j+1}$ is

$$\Gamma_{i,j+1/2}^{n+1} = \left(u_j^{n+1}\right)^+ n_{i,j}^{n+1} + \left(u_j^{n+1}\right)^- n_{i,j+1}^{n+1}.$$

The discrete ion continuity equation is

$$\mathcal{R}_{i,j} = n_{i,j}^{n+1} - n_{i,j}^{n} + \frac{\Delta t}{\Delta x} \left(\Gamma_{i,j+1/2}^{n+1} - \Gamma_{i,j-1/2}^{n+1} \right) - \Delta t S_j = 0.$$

In the expanded implementation, the coefficient multiplying the unsimplified positive/negative velocity expression is

$$C_{i,\mathrm{up}} = \frac{1}{2\Delta x} = 2.54 \times 10^{-1}.$$

The upwind flux introduces numerical diffusion but suppresses nonphysical oscillations in the transported ion density.

Ion momentum equation. The conservative nonlinear momentum flux is approximated by first interpolating the velocity arithmetically:

$$H_{j+1/2}^{n+1} = \frac{1}{2} \left(\frac{u_{i,j}^{n+1} + u_{i,j+1}^{n+1}}{2} \right)^2 = \frac{\left(u_{i,j}^{n+1} + u_{i,j+1}^{n+1}\right)^2}{8}.$$

The ion-momentum residual is

$$\mathcal{R}_{u,j} = u_{i,j}^{n+1} - u_{i,j}^{n} + \Delta t \left[\frac{H_{j+1/2}^{n+1} - H_{j-1/2}^{n+1}}{\Delta x} + \mathcal{A}_i \frac{\phi_{j+1}^{n+1} - \phi_j^{n+1}}{\Delta x} + \widehat{\nu}_i u_{i,j}^{n+1} \right] = 0.$$

The corresponding coefficients appearing directly in the implementation are $C_{u,\mathrm{adv}} = \frac{1}{8\Delta x} = 6.35 \times 10^{-2}$, $C_{u,\phi} = \frac{\mathcal{A}_i}{\Delta x} \approx 1.2253638$, and $C_{u,\nu} = \widehat{\nu}_i = 10^{-2}$. The potential difference $\phi_{j+1} - \phi_j$ is naturally aligned with the velocity index used in the ion mass flux. Under this interface-centered interpretation, the electric field is second-order accurate at the corresponding midpoint. If all variables are instead interpreted as collocated nodal quantities, this term should be regarded as a one-sided difference.

A.7 Nonlinear solution procedure

At every time step, all four fields $\mathbf{z}$ are solved simultaneously as a monolithic nonlinear system,

$$\mathbf{F}\left(\mathbf{z}^{n+1}; \mathbf{z}^{n}, V_{\mathrm{RF}}\right) = \mathbf{0}.$$

No operator splitting is applied between Poisson's equation and the transport equations.

The nonlinear system is solved using `NLsolve.jl`. The first attempt uses Newton's method with a forward-mode automatically differentiated Jacobian:

$$\mathbf{J}\left(\mathbf{z}^{(k)}\right) \delta\mathbf{z}^{(k)} = -\mathbf{F}\left(\mathbf{z}^{(k)}\right), \qquad \mathbf{z}^{(k+1)} = \mathbf{z}^{(k)} + \delta\mathbf{z}^{(k)}.$$

The previous time-step solution is used as the initial guess, which provides a warm start for the nonlinear iteration. The requested function tolerance and maximum number of nonlinear iterations are `ftol` $= 10^{-12}$, and $N_{\mathrm{iter,max}} = 500$. If the Newton solve does not converge according to the `NLsolve.jl` convergence test, the same residual is solved again using a trust-region method with forward-mode automatic differentiation and the same tolerance and iteration limit.

The upwind ion flux contains $|u_i|$ and is therefore only piecewise differentiable at $u_i = 0$. The automatically differentiated residual can still be evaluated, but sign changes near zero velocity may make the local Newton model less smooth. The trust-region fallback provides additional robustness in such cases.

A.8 Positivity treatment

After each nonlinear solve, the electron and ion densities are projected onto the nonnegative set: $n_{e,j} \leftarrow \max(n_{e,j}, 0)$, and $n_{i,j} \leftarrow \max(n_{i,j}, 0)$. This operation is an a posteriori safeguard rather than a positivity-preserving discretization. If clipping occurs at a non-negligible level, the projected state no longer exactly satisfies the nonlinear residual and the projection may perturb discrete particle conservation. Useful diagnostics include the fraction of time steps at which clipping occurs and the normalized clipped mass,

$$\eta_{\mathrm{clip}}^{(s)} = \frac{\sum_j \max\left(-n_{s,j}^{\mathrm{raw}}, 0\right)}{\sum_j \left|n_{s,j}^{\mathrm{raw}}\right| + \varepsilon}, \qquad s \in \{e, i\}.$$

A.9 Stiffness and stability considerations

The electron diffusion term is highly stiff at the selected spatial and temporal resolutions. The dimensionless diffusion number is $r_D = \frac{D_e \Delta t}{\Delta x^2} = \frac{\mathcal{D}_e \Delta \widehat{t}}{\Delta \widehat{x}^2} \approx 3.92257 \times 10^2$. For comparison, the conventional explicit centered diffusion method requires approximately $r_D \leq \frac{1}{2}$. Equivalently, the employed time step is approximately $\frac{\Delta t}{\Delta x^2/(2D_e)} \approx 7.8451 \times 10^2$ times the standard explicit diffusion limit. The fully implicit backward-Euler treatment is therefore essential for avoiding the explicit parabolic time-step restriction.

A.10 Parameter sweep

The sweep varies only the volumetric reaction-source amplitude R. The RF-voltage amplitude is fixed at $V_0 = 100$ V for every trajectory. The source values are linearly spaced according to

$$R_k = R_{\min} + \frac{k}{100}\left(R_{\max} - R_{\min}\right), \qquad k = 0, \ldots, 99,$$

where $R_{\min} = 2.7 \times 10^{18}\ \mathrm{m^{-3}\,s^{-1}}$, and $R_{\max} = 2.7 \times 10^{21}\ \mathrm{m^{-3}\,s^{-1}}$. The trajectories are split into 80 training, 10 validation, and 10 test cases, with the validation and test trajectories selected at approximately uniform intervals over the reaction-source parameter range.

The fixed physical and numerical parameters are summarized in Table 2.

Figure 8 presents representative solution trajectories at low, intermediate, and high reaction-source amplitudes from the parameter sweep. Each panel shows the evolution of one physical variable over the spatial domain and 48 RF cycles, with a common color scale used across the three source amplitudes for each variable. Because the applied voltage is fixed, the electric potential retains a similar RF-periodic pattern throughout the sweep, while the ion velocity preserves its characteristic signed spatial structure. The source amplitude has a stronger influence on the charged-species densities: increasing R_0 substantially raises both n_e and n_i, with their largest values concentrated near the central source region. Across all source amplitudes, the density profiles exhibit slower cycle-to-cycle evolution toward an RF-periodic response, superimposed on the rapid oscillations induced by the boundary forcing. These examples illustrate that the sweep spans distinct plasma-density regimes while preserving the common multiscale structure of the CCP dynamics.

Table 2: Physical and numerical parameters used for every trajectory in the reaction-source sweep.

Quantity	Symbol	Value
Domain length	L	2.5×10^{-2} m
RF voltage amplitude	V_0	100 V
RF frequency	f_{RF}	13.56×10^6 Hz
Source interval	$[x_{s,1}, x_{s,2}]$	$[0.010, 0.015]$ m
Electron-temperature parameter	T_e	3.0
Ion mass	m_i	40 amu
Electron collision frequency	ν_m	10^7 s^{-1}
Ion collision frequency	ν_i	10^6 s^{-1}
Initial density	n_{init}	5×10^{14} m^{-3}
Spatial intervals	N_x	127
Spatial points	N_p	128
Steps per RF cycle	N_t	256
Number of RF cycles	N_c	48
Steps per trajectory	$N_c N_t$	12,288
Nonlinear function tolerance	`ftol`	10^{-12}
Maximum nonlinear iterations	$N_{\mathrm{iter,max}}$	500

B Implementation Details

B.1 Baseline details

FNO For FNOs, we employ four Fourier layers, each retaining 12 Fourier modes, with a latent width of 32 ($k = 12$ and $d_v = 32$). The lifting operator P is implemented as a linear layer, whereas the projection operator Q is represented by a shallow MLP with a single hidden layer with width 128. ReLU activations (Nair & Hinton, 2010) are used throughout. We keep this configuration fixed across all variants to maintain comparable representational capacity, ensuring that performance differences primarily reflect the architectural modifications of each variant rather than variations in these hyperparameters.

FFNO FFNO Tran et al. (2023) factorizes the Fourier kernel into separate one-dimensional operators acting along each coordinate axis. In one-dimensional problems, $D \subset \mathbb{R}$, this factorization reduces to a single Fourier operator along the spatial coordinate. Each FFNO layer then processes the spectral features using a two-layer pointwise channel-mixing network and adds the resulting update through a residual connection:

$$v_{\ell+1}(x) = v_\ell(x) + \sigma\left(W_{\ell,2}\sigma\left(W_{\ell,1}(\mathcal{K}(a;\phi)v_\ell)(x) + b_{\ell,1}\right) + b_{\ell,2}\right).$$

Here, $W_{\ell,1}$ and $W_{\ell,2}$ are pointwise linear transformations, implemented as 1×1 convolutions. Thus, unlike a standard FNO layer that directly combines spectral and pointwise linear updates before applying the activation, the FFNO block transforms the spectral update through a channel-wise MLP and adds it to the latent representation outside the nonlinear mapping. This residual formulation facilitates information and gradient propagation across the stacked operator layers.

Our implementation uses four FFNO residual blocks with independent spectral weights, 12 retained Fourier modes, and a latent width of 32 ($k = 12$ and $d_v = 32$). The lifting operator P is implemented as a linear layer, while the projection operator Q is a shallow MLP with one hidden layer of width 128. ReLU activations are used in both the residual blocks and the projection network. As with the FNO baseline, this configuration is held fixed across the corresponding variants to ensure that performance differences arise primarily from their architectural modifications.

CNO CNO Raonic et al. (2023) employs an encoder–decoder architecture based on local convolutions in the physical domain. Its defining component is an alias-controlled activation that preserves the continuous representation of the input and output functions across different spatial resolutions. In particular, a one-dimensional CNO block maps a latent representation v_ℓ at resolution N_ℓ to $v_{\ell+1}$ at resolution $N_{\ell+1}$ according to

$$v_{\ell+1}(x) = \left[\mathcal{D}_\ell\left(\sigma\left(\mathcal{U}_\ell\left(\mathcal{C}_\ell v_\ell\right)\right)\right)\right](x), \qquad x \in D.$$

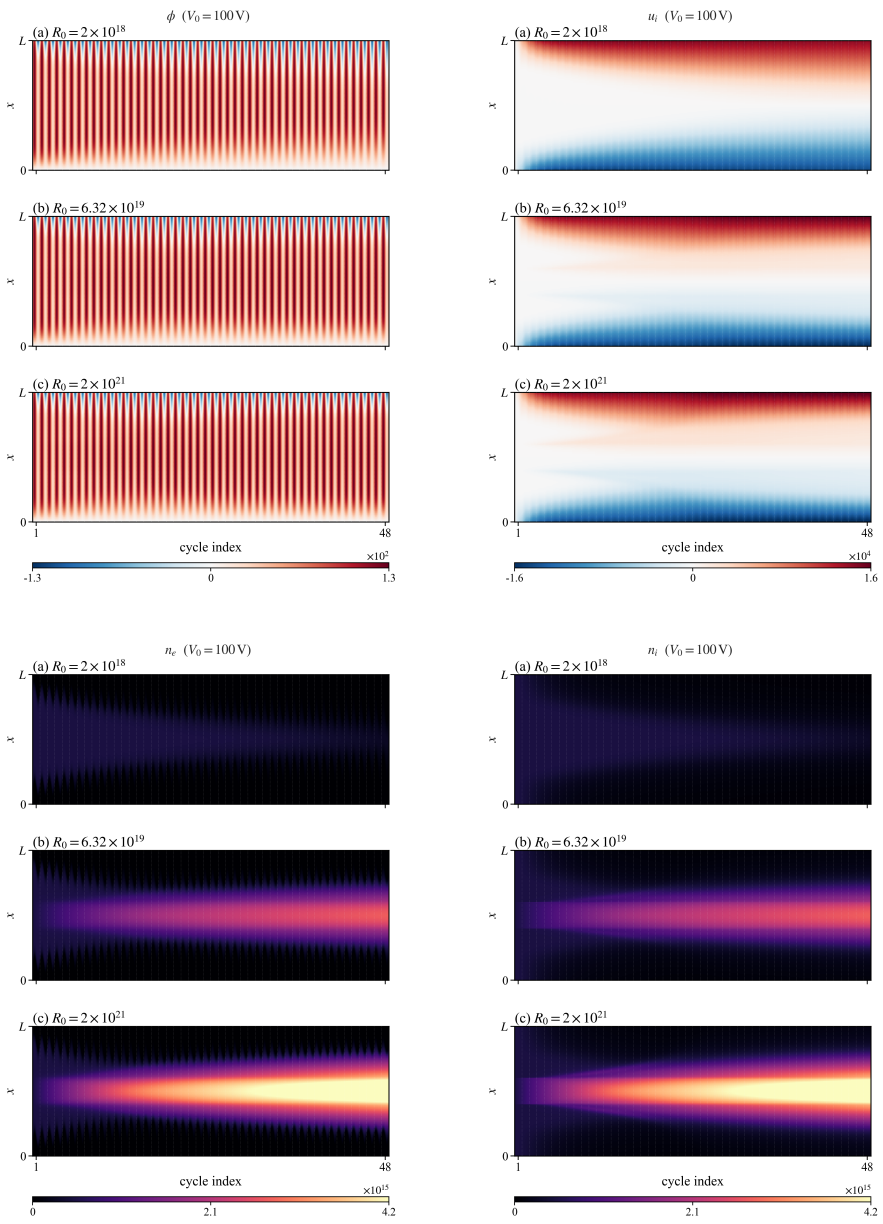


Figure 8: Representative spatiotemporal solution profiles over 48 RF cycles for three reaction-source amplitudes, $R_0 = 2 \times 10^{18}$, 6.32×10^{19}, and $2 \times 10^{21}\,\mathrm{m^{-3}s^{-1}}$, with $V_0 = 100\,\mathrm{V}$. The panels show ϕ, u_i, n_e, and n_i; within each variable, the same color scale is used across the three source amplitudes.

where $\mathcal{C}_\ell$ denotes a learnable one-dimensional convolution, $\mathcal{U}_\ell$ upsamples the convolved features to twice their current spatial resolution, and $\mathcal{D}_\ell$ performs antialiased resampling to the target resolution associated with $v_{\ell+1}$. Applying the nonlinearity σ at the higher intermediate resolution suppresses the aliasing errors that would otherwise arise from a direct pointwise activation. Unlike FNO-based architectures, which parameterize global integral kernels in the Fourier domain, CNO constructs the operator using local physical-space convolutions and multiresolution feature transformations.

These CNO blocks are organized into a U-Net-style architecture. The encoder progressively decreases the spatial resolution while increasing the number of feature channels, whereas the decoder restores

the original resolution using corresponding encoder features through skip connections. Resolution-preserving residual blocks are applied at each encoder level and at the bottleneck to refine the latent features before further downsampling or upsampling.

Our implementation uses two encoder–decoder levels, with four residual blocks at each encoder resolution and four additional residual blocks at the bottleneck. A channel multiplier of 20 produces feature widths of 10, 20, and 40 across the three spatial resolutions, while the lifting and projection blocks use a latent dimension of 32. All convolutions use a kernel size of three. We employ LeakyReLU activations Maas et al. (2013) with a negative slope of 0.01, and batch normalization is disabled throughout the network.

S4FFNO S4FFNO Buitrago Ruiz et al. (2025) augments FFNO with a structured state-space memory layer to represent temporal dependencies that cannot be retained by a purely Markovian one-step operator. Unlike the FNO and FFNO baselines, which concatenate a fixed number of previous solution snapshots along the channel dimension, S4FFNO preserves the temporal ordering of the input sequence and summarizes its history through a recurrent latent state. Let $u_t(x) \in \mathbb{R}^4$ denote the physical state at time step t, and let x denote the spatial coordinate. The S4FFNO update used in this work can be expressed as

$$\begin{aligned} z_t(x) &= (\mathcal{L}_1 \circ \mathcal{L}_0)\left(P\left([u_t(x), x]\right)\right), \\ h_t(x) &= \overline{A} h_{t-1}(x) + \overline{B} z_t(x), \\ m_t(x) &= C h_t(x) + D z_t(x), \\ \widehat{u}_{t+1}(x) &= Q\left((\mathcal{L}_3 \circ \mathcal{L}_2)\left(m_t\right)\right)(x). \end{aligned}$$

Here, P lifts the four physical variables and the spatial coordinate to the latent representation, and $\mathcal{L}_\ell$ denotes the ℓ-th spatial FFNO residual block. The variable $h_t(x)$ is the latent state maintained by the S4D memory layer Gu et al. (2022), while $\overline{A}$ and $\overline{B}$ denote its discretized state-transition and input operators. The matrices C and D map the recurrent state and the current latent input, respectively, to the memory output $m_t(x)$. Finally, Q projects the processed latent representation back to the four physical solution variables.

The S4D recurrence is evaluated independently at each spatial location, with its parameters shared across the spatial domain. Consequently, the FFNO blocks model nonlocal spatial interactions through truncated Fourier operators, whereas the S4D layer models temporal dependencies through its recurrent state. The diagonal structure of the S4D transition operator permits the temporal history to be represented by a compact latent state instead of explicitly concatenating all previous solution snapshots. In our implementation, only the most recent K temporal inputs contribute to the recurrent state, where the default memory-window length is $K = 48$.

The implemented architecture consists of four spatial FFNO residual blocks, with the S4D layer inserted after the second block. Thus, two FFNO blocks process the spatial features before the temporal memory update, and two additional FFNO blocks process the memory-enhanced representation before projection. Each FFNO block retains 12 Fourier modes and uses a latent width of 32 ($k = 12$ and $d_v = 32$). Its pointwise channel-mixing network contains two linear layers with an expansion factor of two and a ReLU activation. The S4D layer uses a state dimension of 64 and is followed by a gated output projection. The lifting operator maps the five input channels, consisting of four physical variables and the spatial coordinate, to the latent width of 32. The projection operator maps the latent representation through a hidden dimension of 128 and returns the four predicted physical variables. Dropout, layer normalization, and weight normalization are not used.

hpFNO$_\mathbf{x}$ For the hpFNO$_x$ baseline, we use the coupled and parameterized FNO architecture described in the main text Jing et al. (2026). The four physical variables are processed within a coupled spectral operator: their Fourier representations are first combined through a mode-wise channel-mixing map, after which a shared Fourier kernel captures cross-variable interactions. Each variable retains a separate pointwise linear transformation in the physical-space branch. Parameter dependence is introduced through a hypernetwork that takes the solution history, spatial coordinate, and normalized reaction-source parameter as inputs and generates additive, layer-dependent shifts for the Fourier layers. The same hypernetwork parameters are used across the four physical variables.

Our implementation employs four coupled Fourier layers, each retaining 12 Fourier modes, with a latent width of 32 ($k = 12$ and $d_v = 32$). A shared linear lifting operator maps the 10-step solution

history, spatial coordinate, and physical parameter to the latent representation. The default linear hypernetwork produces additive shifts for each of the four Fourier layers. The output projection uses separate basis and coefficient maps for each physical variable: each field-specific basis map transforms the latent representation from width 32 to a 128-dimensional basis, and the corresponding coefficient map projects this basis to a scalar output. Layer normalization Ba et al. (2016) is applied jointly to the stacked four-field basis representation before the ReLU activation and final coefficient projection.

hpFNO$_{\mathrm{x}}$-BC This baseline is a boundary-enforced variant of hpFNO$_{\mathrm{x}}$, which is to better represent the RF-driven oscillations of the electric potential. After each one-step prediction, the predicted potential is overwritten at the two boundary points using the prescribed Dirichlet conditions, $\widehat{\phi}(0, t_{j+1}) = 0$ and $\widehat{\phi}(L, t_{j+1}) = V_0 \sin(2\pi f_{\mathrm{RF}} t_{j+1})$. The corrected state is then used to update the input history for the subsequent autoregressive step. This procedure enforces only the known boundary values, while the interior potential remains predicted by the neural operator. In the implementation, the hyper-parameters are set exactly same as the hpFNO$_{\mathrm{x}}$ baseline.

B.2 Proposed method details

hpFNO$_{\mathrm{x}}$-PC Our model adopts the same coupled and parameterized modeling choices as the hpFNO$_{\mathrm{x}}$ architecture. The principal architectural modification is confined to the output projection. Specifically, the model receives the histories of all four physical variables, ϕ, n_e, n_i, and u_i, together with the spatial coordinate and the physical parameter. All four variables therefore participate in the lifting operation, hypernetwork-generated shifts, and coupled spectral interactions throughout the Fourier layers. Because the spectral representations of all four variables are mixed in the hidden representations, it can be interpreted that the electric potential continues to inform the evolution of the three differential variables even though it is not directly predicted by the network.

After the final Fourier layer, only the latent representations associated with n_{e}, n_{i}, and u_{i} are passed to the output projection: from the L hidden representation v_L, the physical quantities are predictied,

$$(\widehat{n}_{\mathrm{e},t+1}, \widehat{n}_{\mathrm{i},t+1}, \widehat{u}_{\mathrm{i},t+1}) = \left(Q_{n_{\mathrm{e}}}\left(v_L^{n_{\mathrm{e}}}\right), Q_{n_{\mathrm{i}}}\left(v_L^{n_{\mathrm{i}}}\right), Q_{u_{\mathrm{i}}}\left(v_L^{u_{\mathrm{i}}}\right)\right).$$

Thus, the change from the original four-output hpFNO$_x$ is implemented by removing the projection associated with v_L^{ϕ}, while leaving the preceding coupled architecture unchanged. The electric potential at the next time step is subsequently reconstructed from the predicted density fields through the Poisson solver,

$$\widehat{\phi}_{t+1} = \mathcal{P}\left(\widehat{n}_{\mathrm{e},t+1}, \widehat{n}_{\mathrm{i},t+1}\right),$$

where $\mathcal{P}$ denotes the numerical Poisson solution operator.

The implementation uses four coupled Fourier layers with 12 retained Fourier modes and a latent width of 32 ($k = 12$ and $d_v = 32$). A shared lifting operator and a shared linear hypernetwork are used across the four input variables, while the pointwise transformations within each Fourier layer remain field-specific. The three predicted variables use separate projection basis and coefficient maps. Each basis map transforms the corresponding 32-dimensional latent representation to a 128-dimensional representation, followed by joint layer normalization, a ReLU activation, and a field-specific scalar coefficient projection.

hpFNO$_{\mathrm{x}}$-FP The hpFNO$_{\mathrm{x}}$-PC model retains the hpFNO$_{\mathrm{x}}$-PC backbone without modifying its local time-stepping architecture. In addition, it introduces an auxiliary RF–Floquet–Poincaré operator, denoted by $\widehat{\mathcal{P}}_{\Theta_{\mathrm{P}}}$, during training. While the hpFNO$_{\mathrm{x}}$-PC backbone approximates the local evolution over a single time step, the auxiliary operator directly approximates the evolution over one complete RF cycle. The two operators are trained jointly so that the local autoregressive trajectory remains consistent with the corresponding one-cycle evolution.

If we define $\mathbf{q}_j(x) = (\phi_j(x), n_{\mathrm{e},j}(x), n_{\mathrm{i},j}(x), u_{\mathrm{i},j}(x))$ as the four-field solution at time index j, and define the delay-embedded input window as $\mathbf{H}_j(x) = [\mathbf{q}_j(x), \ldots, \mathbf{q}_{j+T_{\mathrm{in}}-1}(x)]$, the auxiliary operator learns the one-cycle map

$$\widehat{\mathcal{P}}_{\Theta_{\mathrm{P}}} : (\mathbf{H}_j, x, \mu) \longmapsto \widehat{\mathbf{H}}^{\mathrm{FP},(3)}_{j+P_{\mathrm{RF}}},$$

where P_{RF} is the number of sampled time steps in one RF cycle (P_{RF} =128 in our experiments) and the superscript (3) indicates that the output contains only n_{e}, n_{i}, and u_{i}. Thus, given the four-field history is provided as input, the auxiliary operator directly predicts

$$\widehat{\mathbf{H}}^{\mathrm{FP},(3)}_{j+P_{\mathrm{RF}}} = \left[\widehat{\mathbf{q}}^{\mathrm{FP},(3)}_{j+P_{\mathrm{RF}}}, \ldots, \widehat{\mathbf{q}}^{\mathrm{FP},(3)}_{j+P_{\mathrm{RF}}+T_{\mathrm{in}}-1}\right]. \tag{9}$$

Next we detail the architectural design of the auxiliary operator. Architecturally, the auxiliary operator largely follows the FNO but with several modifications. The auxiliary operator first concatenates the histories of all four variables at each spatial location into a single delay-embedding vector, along with the physical parameter augmented as an extra channel. As in FNO, the lift operator encodes the input into a hidden representation, $z_0(x) = P_{\mathrm{FP}}\left([\mathrm{vec}\left(\mathbf{H}_j(x)\right), x, \mu]\right)$, which is then processed by a sequence of spatial Fourier layers:

$$z_{\ell+1}(x) = \sigma\left(\mathrm{LN}\left(W_\ell z_\ell(x) + \left(\mathcal{K}_\ell z_\ell\right)(x)\right)\right), \qquad \ell = 0, \ldots, L_{\mathrm{FP}} - 1.$$

Here, $\mathcal{K}_\ell$ denotes a one-dimensional spectral convolution, W_ℓ is a pointwise linear transformation, and LN denotes layer normalization. As a nonlinear activation, we use GELU Hendrycks & Gimpel (2016).

The RF–Floquet–Poincaré operator is designed to use a residual parameterization and, thus, the output projection produces a correction

$$\Delta_\Psi(x) = Q_{\mathrm{FP}}\left(z_{L_{\mathrm{FP}}}(x)\right) \in \mathbb{R}^{3\times T_{\mathrm{in}}}.$$

The correction is used to predict the last observed values of the three differential variables in the output window such that:

$$\widehat{\mathbf{H}}^{\mathrm{FP},(3)}_{j+P_{\mathrm{RF}}}(x) = \mathbf{q}^{(3)}_{j+T_{\mathrm{in}}-1}(x)\mathbf{1}^{\mathsf{T}}_{T_{\mathrm{in}}} + \Delta_\Psi(x).$$

As the auxiliary operator targets to approximate convergent cycle-to-cycle operations, the output states can be well approximated by the residual update from the input states. The electric potential is subsequently reconstructed at every predicted time step by applying the numerical Poisson solver:

$$\widehat{\phi}^{\mathrm{FP}}_{j+P_{\mathrm{RF}}+\tau} = \mathcal{S}_{\mathrm{Pois}}\left(\widehat{n}^{\mathrm{FP}}_{\mathrm{e},j+P_{\mathrm{RF}}+\tau}, \widehat{n}^{\mathrm{FP}}_{\mathrm{i},j+P_{\mathrm{RF}}+\tau}; t_{j+P_{\mathrm{RF}}+\tau}\right), \qquad \tau = 0, \ldots, T_{\mathrm{in}} - 1.$$

This produces a four-field, Poisson-consistent target window that can be compared with both the ground-truth trajectory and the local $\mathtt{hpFNO_x\text{-}PC}$ prediction.

The auxiliary operator is trained using a supervised one-cycle prediction loss, minimizing errors in the predicted state Eq. (9):

$$\mathcal{L}_{\mathrm{P}} = \sum_{\tau=0}^{T_{\mathrm{in}}-1} \ell\left(\widehat{\mathbf{q}}^{\mathrm{FP}}_{j+P_{\mathrm{RF}}+\tau}, \mathbf{q}_{j+P_{\mathrm{RF}}+\tau}\right).$$

In addition, the first state predicted by the auxiliary operator is matched with the endpoint obtained from the local $\mathtt{hpFNO_x\text{-}PC}$ rollout:

$$\mathcal{L}_{\mathrm{FP}} = \ell\left(\widehat{\mathbf{q}}^{\mathrm{PC}}_{j+P_{\mathrm{RF}}}, \widehat{\mathbf{q}}^{\mathrm{FP}}_{j+P_{\mathrm{RF}}}\right).$$

The complete training objective is

$$\mathcal{L} = \mathcal{L}_{\Delta t} + \lambda_{\mathrm{P}}\mathcal{L}_{\mathrm{P}} + \lambda_{\mathrm{FP}}\mathcal{L}_{\mathrm{FP}},$$

where $\mathcal{L}_{\Delta t}$ is the standard autoregressive loss of the $\mathtt{hpFNO_x\text{-}PC}$ backbone. We use $\lambda_{\mathrm{P}} = 1$ and $\lambda_{\mathrm{FP}} = 10^{-2}$. The optional cycle-level charge-balance loss is disabled.

For the local $\mathtt{hpFNO_x\text{-}PC}$ backbone, we use four coupled Fourier layers with 12 retained modes and a latent width of 32. As in the $\mathtt{hpFNO_x\text{-}PC}$ configuration, the lifting operator and linear hypernetwork are shared across the four input variables, whereas separate basis and coefficient projections are used for the three predicted variables. Joint layer normalization is applied before their final output projections. The RF–Floquet–Poincaré operator uses four Fourier layers with 12 retained modes and a separate latent width of 64. Its lifting layer maps the $4T_{\mathrm{in}} + 2 = 42$ input features to the latent representation, and its projection network maps the 64-dimensional latent representation through a hidden dimension of 128 to $3T_{\mathrm{in}} = 30$ outputs.

Algorithm 2 Generic Training Algorithm with Random Temporal Subsequence Mini-batches

Input: Trajectory data $\mathbf{U} \in \mathbb{R}^{B\times T\times d}$, trajectory parameters $\{\mu_b\}_{b=1}^{B}$, neural operator $\mathcal{G}_\Theta$, input-history length T_{in}, autoregressive rollout length L_{sub}, number of subsequences N_{b}, number of inner updates N_{in}, number of epochs N_{ep}, loss function ℓ, optimizer OPT

Output: Trained neural operator parameters Θ

for $e = 1, \dots, N_{\text{ep}}$ **do**
  for $m = 1, \dots, N_{\text{in}}$ **do**
    Sample N_{b} temporal windows of length $T_{\text{in}} + L_{\text{sub}}$ according to Algorithm 1
    Denote the sampled batch, trajectory indices, and physical parameters by

$$\mathbf{U}_{\text{sub}} \in \mathbb{R}^{N_{\text{b}}\times(T_{\text{in}}+L_{\text{sub}})\times d}, \quad \mathbf{b} = (b_1, \dots, b_{N_{\text{b}}}), \quad \boldsymbol{\mu}_{\mathbf{b}} = (\mu_{b_1}, \dots, \mu_{b_{N_{\text{b}}}})$$

    Initialize the rollout history using the first T_{in} ground-truth states: $\widehat{\mathbf{H}}^{(0)} \leftarrow (\mathbf{U}_{\text{sub}})_{:,\,1:T_{\text{in}}}$
    Initialize the mini-batch loss: $\mathcal{L}_{\text{sub}} \leftarrow 0$
    for $k = 1, \dots, L_{\text{sub}}$ **do**
      Predict the next state: $\widehat{\mathbf{u}}^{(k)} \leftarrow \mathcal{G}_\Theta(\widehat{\mathbf{H}}^{(k-1)}; \boldsymbol{\mu}_{\mathbf{b}})$
      Compute the prediction loss: $\mathcal{L}_{\text{sub}} \leftarrow \mathcal{L}_{\text{sub}} + \sum_{i=1}^{N_{\text{b}}} \ell(\widehat{\mathbf{u}}_i^{(k)}, \mathbf{u}_i^{(k)})$
      Update the input history: $\widehat{\mathbf{H}}^{(k)} \leftarrow \text{concat}(\widehat{\mathbf{H}}^{(k-1)}_{:,\,2:T_{\text{in}}}, \widehat{\mathbf{u}}^{(k)})$
    end for
    Update the model parameters: $\Theta \leftarrow \text{Opt}(\Theta, \nabla_\Theta \mathcal{L}_{\text{sub}}, \text{optimizer config})$
  end for
end for
return Θ

B.3 Training algorithm

Algorithm 2 summarizes the generic training procedure used for the neural operator models. Following the mini-batch trajectory-training strategy commonly used for continuous-time and Neural ODE-type models (Chen et al., 2018), the optimization is organized using an outer epoch loop and an inner stochastic-update loop. Within each epoch, N_{in} independently sampled mini-batches are processed, and the model parameters are updated once after each mini-batch. Thus, N_{in} denotes the number of stochastic gradient updates performed per epoch.

At each inner iteration, we use Algorithm 1 to sample N_b contiguous temporal windows, setting its generic subsequence-length argument to $T_{\text{in}} + L_{\text{sub}}$. Here, L_{sub} denotes only the autoregressive rollout length: the first T_{in} states of each sampled window initialize the input history, while the remaining L_{sub} states serve as the autoregressive training targets (see Figure 9). Starting from this ground-truth input history, the neural operator repeatedly predicts the next state, and each prediction is appended to the input history used at the subsequent step. Therefore, after initialization, the rollout is performed without teacher forcing.

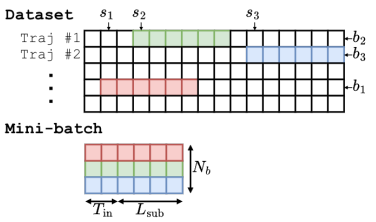


Figure 9: Subsequence sampling

Because temporal windows are generated on the fly, an epoch does not correspond to an exhaustive pass over all possible subsequences. Instead, N_{in} controls the number of newly sampled mini-batches, and hence the computational budget and temporal coverage assigned to each epoch. Over N_{ep} epochs, the model undergoes $N_{\text{ep}}N_{\text{in}}$ parameter updates and processes $N_{\text{ep}}N_{\text{in}}N_{\text{b}}$ sampled temporal windows. This procedure is independent of the particular NO architecture and can therefore be applied consistently to all considered baseline and proposed models.

B.4 Hyperparameters

Unless otherwise specified, all models are trained in double precision using the Adam optimizer Kingma & Ba (2015). The Adam momentum parameters and numerical stability constant are set to their PyTorch default values, $(\beta_1, \beta_2) = (0.9, 0.999)$ and $\epsilon = 10^{-8}$, respectively. A weight

decay of 10^{-4} is applied to the trainable parameters. The initial learning rate is set to $\eta_0 = 2.5 \times 10^{-3}$. The learning rate is reduced by a factor of 0.5 every 100 outer epochs: $\eta_e = \eta_0 \, (0.5)^{\lfloor e/100 \rfloor}$, where e denotes the outer epoch index.

Training is performed for a maximum of $N_{\mathrm{ep}} = 500$ outer epochs. Within each outer epoch, N_{in} randomly resampled mini-batches are processed, and one optimizer update is performed for each mini-batch. Each mini-batch contains $N_{\mathrm{b}} = 10$ temporal windows. A sampled window consists of $T_{\mathrm{in}} = 10$ ground-truth states used to initialize the model and $L_{\mathrm{sub}} = 118$ subsequent states used as autoregressive targets. Consequently, each outer epoch performs nine optimizer updates using a total of 90 sampled temporal windows, and the complete training procedure consists of $N_{\mathrm{ep}} N_{\mathrm{in}} = 4500$ optimizer updates. Gradients are backpropagated through the complete L_{sub}-step autoregressive rollout.

For $\texttt{hpFNO}_{\mathrm{x}}\texttt{-FP}$, the parameters of the $\texttt{hpFNO}_{\mathrm{x}}\texttt{-PC}$ backbone and the auxiliary Floquet–Poincaré operator are optimized jointly using the same optimizer. The loss weights are set to $\lambda_{\mathrm{P}} = 1$ for the supervised Poincaré-map loss and $\lambda_{\mathrm{FP}} = 10^{-2}$ for the Floquet-consistency loss.

At the end of each outer epoch, the model is evaluated by performing a full autoregressive rollout over the validation trajectories. The checkpoint yielding the lowest aggregate validation loss across the four physical variables is retained as the best-performing model. After training, this checkpoint is reloaded and used to obtain the reported results on the held-out test set. The test loss is not used for parameter updates or checkpoint selection.

The backbone architecture and optimization settings of $\texttt{hpFNO}_{\mathrm{x}}\texttt{-PC}$ and $\texttt{hpFNO}_{\mathrm{x}}\texttt{-FP}$ have been inherited directly from $\texttt{hpFNO}_{\mathrm{x}}$ (Jing et al., 2026), without additional tuning. The FP-specific architecture and loss weights have been fixed across all experiments and have not tuned separately for individual experimental settings. For the baseline models, we adopt standard configurations based on the original implementations and have made limited validation-based adjustments, where necessary, to ensure stable training under the common protocol. All hyper-parameter tunings have been performed with a relatively shorter RF cycle setup (i.e., 4 RF cycles) and the same hyper-parameters are used in the longer RF cycles experiments.

Each model is trained using the same five random seeds. Across model families, we observed occasional optimization failures that produced validation errors substantially larger than those of the remaining initializations. To reduce the sensitivity of the aggregate statistics to such failures, we apply the same validation-based upper-tail trimming rule to every baseline and proposed method: for each method, the run with the largest best-achieved validation loss is excluded, and the reported mean and standard deviation are computed over the remaining four runs. This selection is based exclusively on validation performance and is performed without access to the test set.

C Additional Experimental results

C.1 More results on 48 RF cycle trajectories

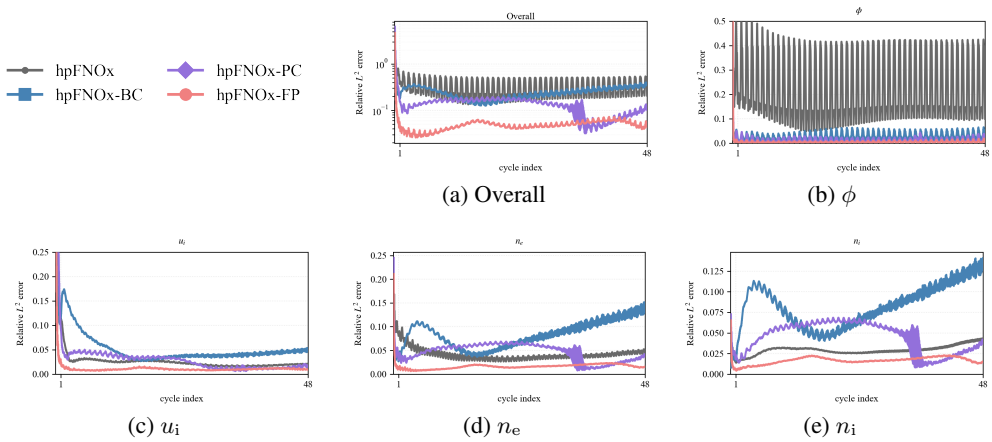
In this section, we present additional results and visualization from the experiments conducted on trajectories with 48 RF cycles, which is the same dataset used in the mainbody of the manuscript.

C.1.1 Time-instantaneous errors

Here we compare the four $\texttt{hpFNO}_{\mathrm{x}}$ variants: $\texttt{hpFNO}_{\mathrm{x}}$, $\texttt{hpFNO}_{\mathrm{x}}\texttt{-BC}$, $\texttt{hpFNO}_{\mathrm{x}}\texttt{-PC}$, and $\texttt{hpFNO}_{\mathrm{x}}\texttt{-FP}$, using time-instantaneous relative ℓ_2 errors. At each time index, the predicted spatial snapshot is compared with the corresponding ground-truth snapshot separately for each physical variable. The overall instantaneous error is obtained by summing the errors of the four physical variables. This evaluation provides a time-resolved view of both the error oscillations within each RF cycle and the gradual accumulation of error across successive cycles.

Figure 10 shows the resulting errors over the 48-RF-cycle rollout. The original $\texttt{hpFNO}_{\mathrm{x}}$ exhibits large periodic errors in the electric potential throughout the trajectory, highlighting the difficulty of directly predicting the strongly oscillatory, boundary-driven field. Enforcing the boundary condition in $\texttt{hpFNO}_{\mathrm{x}}\texttt{-BC}$ substantially reduces the potential error; however, its errors in the electron and ion densities increase during the later cycles, indicating that boundary-condition enforcement alone

is insufficient to control drift in the coupled plasma variables. $\texttt{hpFNO}_\text{x}\texttt{-PC}$ also maintains a low potential error by reconstructing the potential through the Poisson solve, although residual long-horizon errors remain in the evolutionary variables. In comparison, $\texttt{hpFNO}_\text{x}\texttt{-FP}$ maintains the lowest overall error over nearly the entire rollout and consistently achieves low errors across all four physical variables. Its advantage is particularly evident in the electron and ion densities as the rollout proceeds, suggesting that the phase-aligned one-cycle supervision effectively mitigates cycle-to-cycle error accumulation during long-horizon autoregressive prediction.



(a) Overall (b) ϕ

(c) u_i (d) n_e (e) n_i

Figure 10: Time-instantaneous relative ℓ_2 errors over 48 RF cycles for the four $\texttt{hpFNO}_\text{x}$ variants. The overall error is computed as the sum of the four variable-wise errors.

C.1.2 Visualization of Errors

Figure 11 provides a spatiotemporal view of the prediction errors for the four $\texttt{hpFNO}_\text{x}$ variants. Starting from the same 10-step ground-truth history, each model is rolled out autoregressively over the remaining prediction horizon. For each method, the predictions from $N_\text{ens} = 4$ independently trained models are first decoded to the original physical scale and averaged to form an ensemble prediction. The pointwise absolute error of this ensemble prediction is then averaged over $N_\text{test} = 10$ held-out test trajectories. Specifically, for each physical variable $q \in \{\phi, n_e, n_i, u_i\}$, the displayed error is

$$E_q(x,t) = \frac{1}{N_\text{test}} \sum_{j=1}^{N_\text{test}} \left| \frac{1}{N_\text{ens}} \sum_{m=1}^{N_\text{ens}} \widehat{q}_m^{(j)}(x,t) - q^{(j)}(x,t) \right|,$$

where $\widehat{q}_m^{(j)}$ denotes the prediction from the m-th independently trained model for the j-th test trajectory. The horizontal axis is reported in RF-cycle index, with 128 retained time steps per cycle, while the vertical axis represents the spatial coordinate $x \in [0, L]$. All four methods within each row share the same color scale, allowing direct comparison across methods for a given physical variable. Because each variable is shown in its own physical units and with a separate color scale, color intensities should not be compared across different rows.

The electric-potential error exhibits the clearest distinction between the unconstrained and physics-constrained models. In $\texttt{hpFNO}_\text{x}$, the error remains strongly periodic over the entire rollout and is most pronounced near the driven boundary $x = L$, reflecting persistent inaccuracies in representing the RF-conditioned potential. Enforcing the prescribed boundary values in $\texttt{hpFNO}_\text{x}\texttt{-BC}$ substantially suppresses this error, while both $\texttt{hpFNO}_\text{x}\texttt{-PC}$ and $\texttt{hpFNO}_\text{x}\texttt{-FP}$ maintain uniformly low potential errors by reconstructing the complete potential field through the Poisson solve.

The error distributions of the evolutionary variables reveal an additional distinction among the constrained models. Although $\texttt{hpFNO}_\text{x}\texttt{-BC}$ accurately imposes the potential boundary values, its errors in n_e and n_i become increasingly pronounced within the interior of the domain as the rollout proceeds, and its u_i error remains concentrated near the two boundaries. $\texttt{hpFNO}_\text{x}\texttt{-PC}$ reduces the density errors relative to the boundary-condition-only model, but persistent error bands remain in the

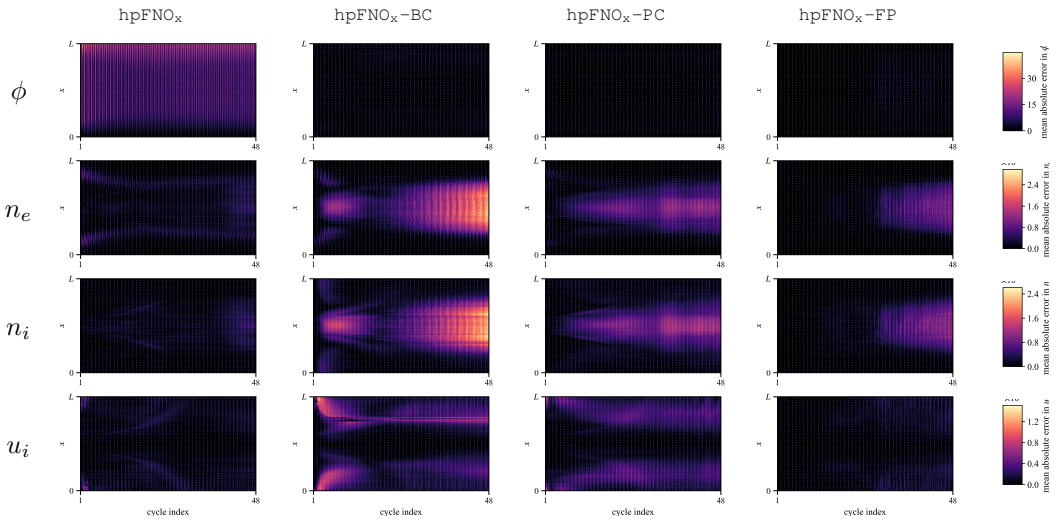


Figure 11: Spatiotemporal mean absolute errors over 48 RF cycles, computed from four-model ensemble predictions and averaged over 10 test trajectories. Methods within each row share the same color scale.

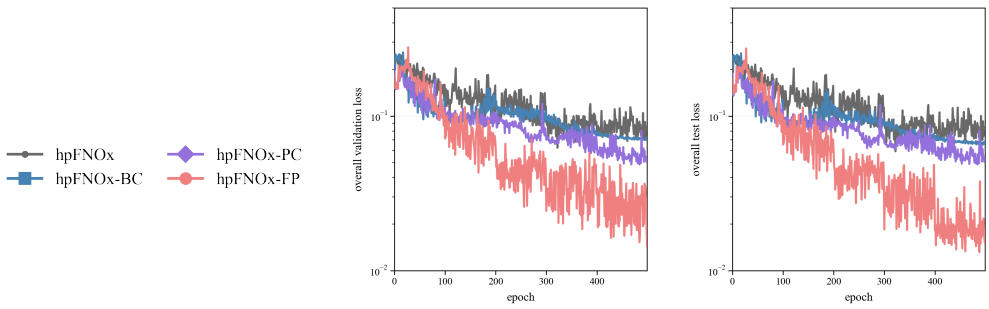
density and velocity fields. In comparison, $\texttt{hpFNO}_\text{x}\texttt{−FP}$ keeps the errors small over a larger portion of the rollout and confines the remaining late-cycle density errors to a narrower spatial region. It also produces a substantially lower and more spatially uniform ion-velocity error. Since $\texttt{hpFNO}_\text{x}\texttt{−PC}$ and $\texttt{hpFNO}_\text{x}\texttt{−FP}$ employ the same Poisson closure, their similar potential-error maps but markedly different errors in n_e, n_i, and u_i isolate the benefit of the phase-aligned cycle-to-cycle regularization. Overall, the FP formulation provides the most balanced spatiotemporal accuracy across the algebraic and evolutionary variables during the long autoregressive rollout.

C.1.3 Validation and test loss trajectories

To examine how predictive performance evolves throughout training, we evaluate each model checkpoint on the held-out validation and test sets after every training epoch. Figure 12 reports the resulting overall validation and test losses on a logarithmic scale, averaged over four random seeds. The two evaluation splits exhibit closely matched trends and nearly identical relative rankings among the methods, indicating that the validation trajectories provide a representative view of generalization to the held-out test cases. All four $\texttt{hpFNO}_\text{x}$ variants exhibit a rapid reduction in loss during the initial training stage. As training proceeds, however, their trajectories become more separated. The Poisson-closed model, $\texttt{hpFNO}_\text{x}\texttt{−PC}$, generally attains lower late-stage losses than $\texttt{hpFNO}_\text{x}$ and $\texttt{hpFNO}_\text{x}\texttt{−BC}$, while $\texttt{hpFNO}_\text{x}\texttt{−FP}$ exhibits the most substantial and sustained improvement. In particular, $\texttt{hpFNO}_\text{x}\texttt{−FP}$ continues to reduce both validation and test losses after the competing methods enter a slower-improvement regime, ultimately achieving the lowest losses by a clear margin. Despite some epoch-to-epoch variability, its overall downward trend is consistently observed on both data splits, suggesting that the phase-aligned one-cycle supervision provides an informative training signal beyond learning only the local Δt-scale dynamics. The absence of a late-stage divergence between its validation and test trajectories further indicates that the improvement transfers to unseen physical-parameter instances rather than being restricted to the validation set.

C.1.4 Temporal extrapolation setup

While long-horizon extrapolation is not the primary focus of this work, we include an additional temporal extrapolation experiment as a diagnostic stress test to assess whether the learned dynamics remain stable and predictive beyond the time horizon encountered during training. Importantly, we do not introduce any extrapolation-specific training objective, rollout curriculum, stability regularization, fine-tuning, or state-correction mechanism for this experiment.



(a) Validation loss (b) Test loss

Figure 12: Overall validation and test losses monitored after each training epoch for the four $\texttt{hpFNO}_\text{x}$ variants.

All models are trained using trajectories spanning the first 48 RF cycles, whereas the corresponding test trajectories are extended to 64 RF cycles. For each test trajectory, the model is initialized using the same (T_in)-snapshot input window at the beginning of the trajectory and then advanced through a single autoregressive rollout to the end of the 64th RF cycle. The rollout is therefore not restarted from the ground-truth solution at cycle 48, and no reference states are supplied beyond the initial input window. The normalization statistics are likewise computed using only the training trajectories over the original 48-cycle horizon, preventing information from the extended test horizon from entering either the model or the preprocessing procedure.

Because prediction accuracy over RF cycles 1–48 is reported in the main text, we evaluate temporal extrapolation exclusively over cycles 49–64. These 16 additional cycles extend the prediction horizon by approximately (33%) beyond that observed during training, and their ground-truth solutions are used only for error evaluation. The resulting performance should therefore be interpreted as an assessment of the models' zero-shot temporal robustness rather than as the best achievable performance of a method specifically optimized for long-horizon extrapolation.

Table 3: Performance comparison (values are scaled by 10^{-1} and formatted as mean ($\pm$ standard deviation)): A long rollout experiment with 64RF cycle trajectories (8192 temporal steps). The individual prediction errors are measured for each physical variable (ϕ, n_e, n_i, u_i) and the overall performance (i.e., the sum of all nRMSEs).

Method	Overall	ϕ	n_e	n_i	u_i
FNO	17.642 ($\pm$ 2.3600)	3.4864 ($\pm$ 2.1139)	6.6075 ($\pm$ 0.2112)	6.2804 ($\pm$ 0.1652)	1.2674 ($\pm$ 0.0445)
FFNO	21.262 ($\pm$ 1.9153)	5.9523 ($\pm$ 1.4197)	7.1824 ($\pm$ 0.8485)	6.8093 ($\pm$ 0.7048)	1.3176 ($\pm$ 0.1952)
CNO	20.771 ($\pm$ 0.2050)	5.2811 ($\pm$ 0.0643)	7.2169 ($\pm$ 0.2951)	6.8798 ($\pm$ 0.1726)	1.3928 ($\pm$ 0.2106)
S4FFNO	15.781 ($\pm$ 2.7187)	1.9042 ($\pm$ 2.1003)	6.6438 ($\pm$ 0.6224)	6.2811 ($\pm$ 0.5874)	0.9521 ($\pm$ 0.2711)
DISCO	46.030 ($\pm$ 5.5881)	9.6759 ($\pm$ 1.5800)	15.260 ($\pm$ 5.0880)	11.624 ($\pm$ 4.1833)	9.4702 ($\pm$ 4.4703)
$\texttt{hpFNO}_\text{x}$	3.6133 ($\pm$ 2.7080)	1.9018 ($\pm$ 2.2676)	0.7641 ($\pm$ 0.2788)	0.6807 ($\pm$ 0.1821)	0.2667 ($\pm$ 0.1198)
$\texttt{hpFNO}_\text{x}$-BC	4.3094 ($\pm$ 3.9279)	0.2881 ($\pm$ 0.3018)	1.7624 ($\pm$ 1.6098)	1.6710 ($\pm$ 1.4749)	0.5879 ($\pm$ 0.5446)
$\texttt{hpFNO}_\text{x}$-PC	3.1689 ($\pm$ 2.8972)	0.3598 ($\pm$ 0.3724)	1.1749 ($\pm$ 1.0391)	1.1241 ($\pm$ 0.9986)	0.5102 ($\pm$ 0.4911)
$\texttt{hpFNO}_\text{x}$-FP	1.7108 ($\pm$ 0.6639)	0.1364 ($\pm$ 0.1108)	0.6895 ($\pm$ 0.2434)	0.6480 ($\pm$ 0.2267)	0.2369 ($\pm$ 0.0973)

Table 3 reports temporal extrapolation errors over RF cycles 49–64. The $\texttt{hpFNO}_\text{x}$ variants substantially outperform the completed conventional baselines, with $\texttt{hpFNO}_\text{x}$-FP achieving the lowest mean error for every physical variable. Its overall error is approximately 89.2% lower than that of S4FFNO and 46.0% lower than that of $\texttt{hpFNO}_\text{x}$-PC, suggesting that the one-cycle consistency objective helps suppress long-term error accumulation.

C.2 DETAILED RESULTS ON 4 RF CYCLE TRAJECTORIES

Table 4 presents the prediction accuracy, measured by nRMSE, for the baseline models (FNO, FFNO, CNO, S4FFNO, DISCO, hpFNO$_x$, and hpFNO$_x$−BC) and the proposed hpFNO$_x$−PC and hpFNO$_x$−FP methods. All models are trained using the same protocol, with $T_{\text{in}} = 10$, $L_{\text{sub}} = 118$, $N_{\text{b}} = 10$, and $N_{\text{in}} = 9$. Among the baselines, the hpFNO$_x$ variants consistently outperform the general-purpose neural operators. In particular, enforcing the prescribed potential boundary conditions in hpFNO$_x$−BC substantially improves the prediction of the oscillatory CCP dynamics. hpFNO$_x$−PC achieves the lowest overall error, outperforming all considered baselines. Although hpFNO$_x$−FP remains more accurate than the original hpFNO$_x$ and the general-purpose baselines, it does not improve upon hpFNO$_x$−PC in this short-horizon setting. This result is consistent with the role of FP regularization: four-cycle trajectories contain relatively few distinct cycle-to-cycle transitions and provide limited opportunity for long-term rollout drift to accumulate. The auxiliary one-cycle supervision may therefore be less informative in this setting, whereas its benefit becomes more pronounced for the 48-cycle trajectories considered in the main experiments.

Table 4: Performance comparison (values are scaled by 10^{-1} and formatted as mean (± standard deviation)): A rollout experiment with 4RF cycle trajectories (512 temporal steps). The individual prediction errors are measured for each physical variable (ϕ, n_{e}, n_{i}, u_{i}) and the overall performance (i.e., the sum of all nRMSEs).

Method	Overall	ϕ	n_e	n_i	u_i
FNO	2.0261 (± 0.0665)	0.4586 (± 0.0372)	0.7412 (± 0.0273)	0.6401 (± 0.0213)	0.1862 (± 0.0199)
FFNO	2.3055 (± 0.4674)	0.5471 (± 0.3397)	0.8066 (± 0.0385)	0.7050 (± 0.0256)	0.2468 (± 0.0769)
CNO	6.2712 (± 1.3068)	3.8736 (± 0.9203)	1.1937 (± 0.2551)	0.7349 (± 0.2304)	0.4691 (± 0.1037)
S4FFNO	1.2989 (± 0.2091)	0.3082 (± 0.0197)	0.4800 (± 0.1025)	0.4095 (± 0.1155)	0.1012 (± 0.0100)
DISCO	2.0644 (± 0.4896)	0.7252 (± 0.1774)	0.5734 (± 0.1050)	0.3040 (± 0.1071)	0.4618 (± 0.1154)
hpFNO$_x$	0.1870 (± 0.0194)	0.0558 (± 0.0068)	0.0712 (± 0.0129)	0.0164 (± 0.0014)	0.0437 (± 0.0071)
hpFNO$_x$−BC	0.1326 (± 0.0104)	0.0333 (± 0.0033)	0.0467 (± 0.0110)	0.0185 (± 0.0038)	0.0341 (± 0.0032)
hpFNO$_x$−PC	0.1086 (± 0.0100)	0.0185 (± 0.0038)	0.0320 (± 0.0059)	0.0170 (± 0.0024)	0.0410 (± 0.0119)
hpFNO$_x$−FP	0.1563 (± 0.0443)	0.0206 (± 0.0043)	0.0396 (± 0.0095)	0.0271 (± 0.0078)	0.0690 (± 0.0262)

C.3 EXPERIMENTS ON VOLTAGE PARAMETER SWEEP

We further evaluate the parameter-generalization capability of the four hpFNO$_x$ variants by varying the amplitude of the applied RF voltage. The reaction-source coefficient is fixed at $R_0 = 2.7 \times 10^{19}$, while the voltage amplitude is uniformly sampled as

$$V_0^{(k)} = 100 + 2k \text{ V}, \qquad k = 0, \ldots, 99,$$

resulting in 100 operating conditions spanning $V_0 \in [100, 300]$ V. Like the reaction coefficient sweep case, the trajectories are split into 80 training, 10 validation, and 10 test cases, with the validation and test trajectories selected at approximately uniform intervals over the voltage parameter range. For each voltage, we use a trajectory consisting of four consecutive RF cycles, while retaining the same spatial discretization, temporal sampling, input-history length, and model configurations used in the preceding experiments.

Unlike the reaction-source sweep, varying V_0 directly changes the time-dependent Dirichlet boundary condition,

$$\phi(L, t) = V_0 \sin(2\pi f_{\text{RF}} t),$$

and consequently alters the electric field, sheath dynamics, charged-particle densities, and ion velocity. This experiment therefore evaluates whether the models can generalize across parameter-dependent RF excitations rather than only across variations in an interior source term. The quantitative results for hpFNO$_x$, hpFNO$_x$−BC, hpFNO$_x$−PC, and hpFNO$_x$−FP are reported in Table 5.

Table 5: Varying voltage experiment: Performance comparison (values are scaled by 10^{-1} and formatted as mean ($\pm$ standard deviation)): A rollout experiment with 4RF cycle trajectories (512 temporal steps). The individual prediction errors are measured for each physical variable (ϕ, n_{e}, n_{i}, u_{i}) and the overall performance (i.e., the sum of all nRMSEs).

Method	Overall	ϕ	n_e	n_i	u_i
FNO	1.7683 ($\pm$ 0.2260)	0.8884 ($\pm$ 0.1663)	0.4535 ($\pm$ 0.0502)	0.0580 ($\pm$ 0.0105)	0.3683 ($\pm$ 0.0349)
FFNO	1.5246 ($\pm$ 0.2120)	0.7160 ($\pm$ 0.1172)	0.4234 ($\pm$ 0.0628)	0.0425 ($\pm$ 0.0066)	0.3428 ($\pm$ 0.0483)
CNO	4.1350 ($\pm$ 0.9187)	1.9140 ($\pm$ 0.4629)	1.0944 ($\pm$ 0.2543)	0.1586 ($\pm$ 0.0542)	0.9680 ($\pm$ 0.2611)
S4FFNO	1.7850 ($\pm$ 0.5753)	0.9167 ($\pm$ 0.2227)	0.4423 ($\pm$ 0.1375)	0.0441 ($\pm$ 0.0202)	0.3820 ($\pm$ 0.1970)
DISCO	4.4692 ($\pm$ 2.0798)	2.0084 ($\pm$ 0.8701)	1.3407 ($\pm$ 0.4794)	0.1666 ($\pm$ 0.0988)	0.9536 ($\pm$ 0.6530)
hpFNO$_{\mathrm{x}}$	0.3123 ($\pm$ 0.0705)	0.1340 ($\pm$ 0.0367)	0.1175 ($\pm$ 0.0304)	0.0116 ($\pm$ 0.0022)	0.0491 ($\pm$ 0.0067)
hpFNO$_{\mathrm{x}}$-BC	0.1444 ($\pm$ 0.0024)	0.0462 ($\pm$ 0.0011)	0.0557 ($\pm$ 0.0038)	0.0094 ($\pm$ 0.0023)	0.0331 ($\pm$ 0.0027)
hpFNO$_{\mathrm{x}}$-PC	0.1186 ($\pm$ 0.0089)	0.0382 ($\pm$ 0.0023)	0.0377 ($\pm$ 0.0030)	0.0086 ($\pm$ 0.0016)	0.0341 ($\pm$ 0.0042)
hpFNO$_{\mathrm{x}}$-FP	0.1280 ($\pm$ 0.0050)	0.0371 ($\pm$ 0.0014)	0.0396 ($\pm$ 0.0039)	0.0098 ($\pm$ 0.0015)	0.0415 ($\pm$ 0.0068)

D LIMITATIONS.

The Poisson closure requires an additional numerical solve at every autoregressive step. During end-to-end training, gradients must also propagate through this solve, increasing the computational cost; while the fixed one-dimensional system considered here admits an efficient factorized solve, this overhead may become more substantial in higher-dimensional or irregular domains. The current finite-difference closure is also tied to a particular spatial discretization and geometry, such that transfer across resolutions or meshes requires reconstructing the discrete Poisson operator and its associated solver. In terms of model complexity, hpFNO$_{\mathrm{x}}$-PC has slightly fewer learnable parameters than hpFNO$_{\mathrm{x}}$ because the potential output projection is removed. However, hpFNO$_{\mathrm{x}}$-FP introduces an auxiliary one-cycle operator, increasing the training time parameter count, memory footprint, and optimization cost. Since this auxiliary operator is discarded after training, the additional model complexity does not carry over to inference. Finally, our experiments are restricted to a one-dimensional geometry, leaving broader operating conditions and higher dimensional CCP models for future work.