Sparsified Time-Dependent Fourier Neural Operators for Fusion Simulations

Mustafa Mutiur Rahman,[1] Zhe Bai,[1] Jacob Robert King,[2] Carl R. Sovinec,[3] Xishuo Wei,[4] Samuel Williams,[1] and Yang Liu[1]

[1)] *Lawrence Berkeley National Laboratory, Berkeley, CA, 94720*
[2)] *Fiat Lux, Lafayette, CO 80026*
[3)] *University of Wisconsin-Madison, Madison, WI, 53706*
[4)] *University of California, Irvine, CA, 92617*

(*Electronic mail: liuyangzhuan@lbl.gov.)

(Dated: 28 October 2024)

This paper presents a sparsified Fourier neural operator for coupled time-dependent partial differential equations (ST-FNO) as an efficient machine learning surrogate for fluid and particle-based fusion codes such as NIMROD and GTC. ST-FNO leverages the structures in the governing equations and utilizes neural operators to represent Green's function-like numerical operators in the corresponding numerical solvers. Once trained, ST-FNO can rapidly and accurately predict dynamics in fusion devices compared with first-principle numerical algorithms. In general, ST-FNO represents an efficient and accurate machine learning surrogate for numerical simulators for multi-variable nonlinear time-dependent partial differential equations, with the proposed architectures and loss functions. The efficacy of ST-FNO has been demonstrated using quiescent H-mode simulation data from NIMROD and kink-mode simulation data from GTC. The ST-FNO H-mode results show orders of magnitude reduction in memory and CPU usage in comparison with the numerical solvers in NIMROD when computing fields over a selected poloidal plane. The ST-FNO kink-mode results achieve a factor of 2 reduction in the number of parameters compared to baseline FNO models without accuracy loss.

I. INTRODUCTION

Simulation codes are indispensable tools in fusion plasma confinement research, enabling the study of plasma instabilities, prediction of plasma behaviors, design and control of fusion devices, and other critical aspects contributing to the advancement of fusion energy. Mainstream fusion codes use numerical approximations to solve physical models of varying fidelity and applicability with respect to temporal and spatial scales. For example, nonlinear extended magneto-hydrodynamics (MHD) codes are routinely applied to study macroscale instability and its consequences in a broad class of magnetic fusion energy (MFE) devices such as tokamaks, stellarators, reversed-field pinches, and field-reversed configurations. Among many others, they include codes like NIMROD,[1] M3D-C1,[2] and JOREK.[3,4] Understanding cross-field transport due to microscale turbulence in MFE configurations requires a description of the kinetic deviations from a Maxwellian distribution function, due to important effects at the scale of particle gyroradii. Here, the particle gyro-averaged kinetic model, gyrokinetics (GK), is solved in both Eulerian form by codes such as GENE[5] and GYRO[6] and in Lagrangian particle-in-cell (PIC) form by codes like GTC [7,8] and XGC[9]. Despite their success, these "first-principles" fusion codes are computationally demanding for high-fidelity and whole-device simulation, and each practical run takes days using thousands of CPUs or GPUs on the leadership supercomputers.

Data-driven algorithms, particularly scientific machine learning (SciML) models for partial differential equations (PDEs), offer appealing complements and alternatives to the fusion codes, as they require much less computation and memory resources to predict an approximate solution once the models have been trained. Compared to other data-driven algorithms, SciML models incorporate physics knowledge and principles into the neural network design to reduce model sizes, improve training time, reduce training samples, and/or improve model accuracy. SciML models can potentially be used for fusion research in the following ways: (1) They can be used for the diagnosis of critical events in the fusion device by analyzing the snapshots generated by simulation and experiments. (2) They can be used to quickly predict long-range dynamics without running the simulation code with small time step sizes. (3) They can be used to produce initial guesses and preconditioners for solving linear systems arising in certain fusion codes. (4) They also allow training the ML model with one set of initial conditions or physics models, and testing the model with another set of initial conditions or physics models. Existing SciML models include physics-informed neural networks (PINN) that encode PDEs into the loss function[10], DeepOnet[11], Fourier neural operators (FNO)[12] for operator learning, Gaussian process[13,14], and reduced-order models[15,16], to name a few. To date, most SciML models have been demonstrated in simplified configurations, and very few have been considered for production simulation in realistic geometry and with experimental data[17–25]. In this paper, we aim at developing new SciML models for production fusion computations based on the FNO model.

Inspired by the pseudo-spectral method,[26] the FNO model represents Green's functions in the spectral domain by nonlinear network operators[12]. Similar to the dealiased pseudo-spectral method that zeros out higher-frequency components, for example see Ref. 27, the FNO model keeps only the lowest few modes for efficiency and accuracy to achieve superior performance for PDEs with smooth solutions. FNO has been mainly applied to relatively simple time-dependent fluid dynamics simulations to predict solutions at future time

from their past snapshots. We note that the FNO model is also well-suited for augmenting fusion codes, as many exploit Fourier transforms for periodic directions. Another consideration is that linearly unstable perturbations often exhibit alignment with the magnetic field, making the Fourier representation with field-line-following coordinates a compact representation for GK computation. Although FNO is parameter-efficient, further reducing the model size is critical for large-scale production fusion simulations, particularly due to the limited memory capacity of GPU devices. In addition to the baseline FNO model[12], there exist a few FNO variants including Tensorized FNO (TFNO)[28] which leverages tensor compression of the network parameters, geo-FNO[29] that adapts FNO to arbitrary geometry and mesh, factorized FNO[30] that simplifies the FNO operations, and physics-informed FNOs[31,32] that improve the model accuracy by incorporating PDE-based residuals.

Aside from these improved models, we remark that there is another largely open opportunity to improve FNO for fusion codes: production codes often solve a coupled system of equations involving a handful of variables such as temperature, pressure, magnetic field, current density, flow velocity, electrostatic and magnetic potentials, etc. One can leverage the sparse dependencies of these quantities, as indicated by the governing equations, to further simplify the FNO model architecture. We exploit this idea in this paper for two challenging applications of fusion codes that stretch their typical use. Our extended-MHD application models profile evolution from edge-driven fluctuations using the two-fluid description to reproduce electron-fluid dynamics that are not represented by MHD. Conversely, our GK application models macroscale evolution including kinetic effects over a relatively long time.

Our contribution is three-fold:

- We bridge the gap between advanced SciML models with production fusion codes (those that solve multiple equations with large-scale and unstructured mesh). Our general methodology applies to fusion codes with linear and nonlinear mode modeling capabilities.
- We propose a parameter-efficient SciML model called sparsified FNO for coupled time-dependent partial differential equations (ST-FNO) that leverages the structure of the governing equations to simplify the architecture. ST-FNO can achieve up to 2x parameter reduction while maintaining similar inference accuracy as the baseline FNO model.
- We demonstrate the efficiency and accuracy of ST-FNO with two distinct fusion codes: NIMROD (extended-MHD) for a simulation of a saturated tokamak edge perturbation, and GTC (gyrokinetic PIC) for a kink mode simulation. Compared with the algorithm in NIMROD, which uses SuperLU_DIST to precondition the computation-intensive algebraic solves, ST-FNO can achieve over 100x memory reduction and significant speedup for generating approximate field solutions over a single poloidal plane, once the model has been trained.

II. METHODOLOGY

We first give a brief introduction of FNO and its building blocks in Section II A. We then explain in detail how to develop the proposed ST-FNO frameworks for two distinct examples of fusion simulation codes: NIMROD (Section II B) and GTC (Section II C) by essentially identifying the independent field variables, their governing equations and sparsifying connectivity in FNO. Finally, we present a short remark on how to develop ST-FNO for other fusion codes and, more generally, time-dependent PDEs in Section II D.

A. FNO

FNO[12] represents, arguably, one of the most successful machine learning tools for PDE simulations, and has been demonstrated on a variety of simple and production fluid dynamics simulation codes[17,31]. FNO is an operator learning framework that represents the discretized Green's function in the spectral domain with parameter efficient neural network components. Suppose one wants to use FNO to predict a set of n_F physical quantities F_s, $s = 1,\ldots,n_F$ temporally discretized at time steps $k+1,\ldots,k+k_o$ from their past values at time steps $k-k_i+1,\ldots,k$. Assume that each physical quantity is spatially discretized into a $n_1 \times n_2$ array. In other words, the input of FNO for each physical quantity is of dimensions $n_1 \times n_2 \times k_i$ and the output is of dimensions $n_1 \times n_2 \times k_o$.

A typical FNO architecture consists of a lifting operator P_m, several FNO layers with FNO operator $\mathrm{F}_{m,m}$ and a projection operator Q_m. Here, m denotes the number of input and output physical quantities of a FNO operator. In our baseline model, it is assumed that $m = n_F$, and the input and output numbers of physical quantities are the same. But as will be seen later, our proposed FNO model builds upon F_{m_i,m_o} with $m_i, m_o \leq n_F$. Figs. 1(a) and 1(b) show the baseline FNO models (similar to the idea of[24]) for the NIMROD and GTC cases (see Sections II B and II C for detailed explanation). The lifting operator P_m pads the input array with the normalized (x,y) spatial coordinates into a $n_1 \times n_2 \times (mk_i+2)$ array and lifts the size of the last dimension to mW via linear transformation (see Fig. 1(c)). W represents the internal width of each field quantity. Each FNO layer consists of one FNO operator $\mathrm{F}_{m,m}$ (see Fig. 1(d)) that performs fast Fourier transforms (FFT) along the x and y dimensions and only keeps the lowest M Fourier components. These components are passed to a convolution operator R which learns the underlying Green's function of the PDE. Afterwards, they are converted back to data of dimensions $n_1 \times n_2 \times nW$ via inverse fast Fourier transforms (IFFT). In addition, a convolution operator W with kernel size 1 is also used to regularize the learning process (see Refs. 12 and 24 for more explanation). Finally, the projection operator P_m converts the data to the output of dimensions $n_1 \times n_2 \times k_o$ (see Fig. 1(c)). In $\mathrm{F}_{m,m}$ and Q_m, the GeLU activation function is used.

Among the operators P_m, Q_m and $\mathrm{F}_{m,m}$, the parameter count of FNO is dominated by that in $\mathrm{F}_{m,m}$, each containing roughly $m^2W^2M^2$ parameters. This number is estimated by the fact

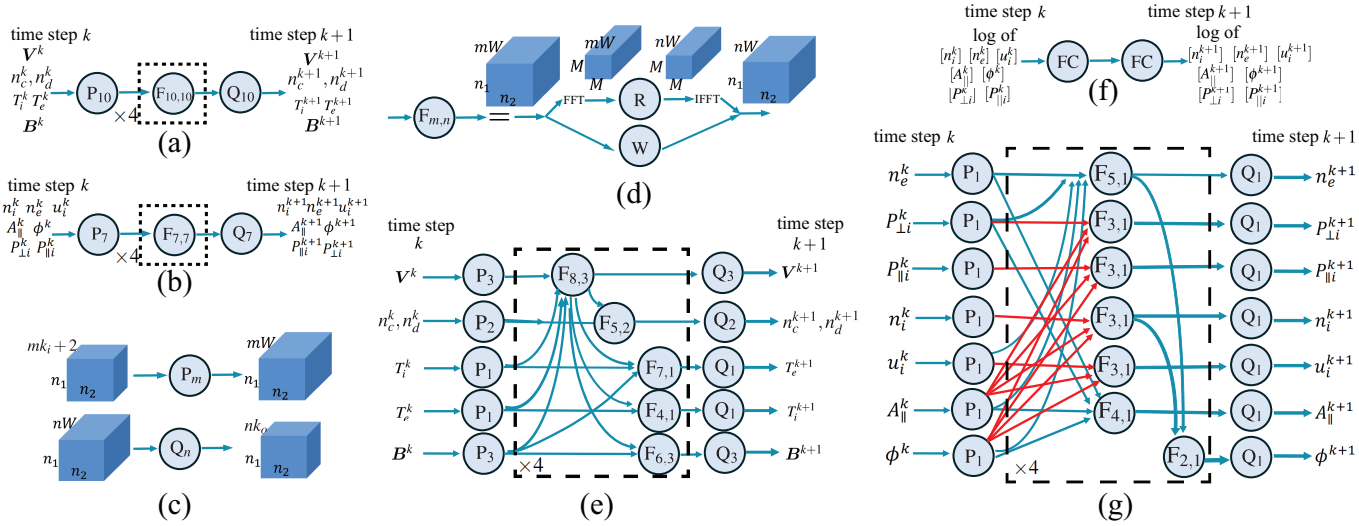

FIG. 1. (a) Baseline-FNO model with $k_i = 1$, $k_o = 1$ for the NIMROD case, "$\times 4$" represents 4 Fourier layers $\mathrm{F}_{m,n}$. (b) Baseline-FNO model with $k_o = 1$, $k_o = 1$ for the GTC case. (c) Input and output dimensions for the lifting operator P_m and projection operator Q_n. (d) Data dimensions for the Fourier layer $\mathrm{F}_{m,n}$. (e) Proposed ST-FNO model with $k_i = 1$, $k_o = 1$ for the NIMROD case. (f) The RMS-FC model to predict the magnitude information for the GTC case. (g) Proposed ST-FNO model with $k_i = 1$, $k_o = 1$ for the GTC case.

that (1) the FNO operator is inspired by the pseudo-spectral method, which decouples the interactions among Fourier modes, and (2) the R operator performs convolution along the mW dimension. Therefore, the parameter count is linear with respect to the total number of retained Fourier modes M^2 and quadratic with respect to the total internal width mW. It's critical to choose the proper value for the internal width W and the number of Fourier modes M in each spatial coordinate to construct an efficient and effective FNO model. That said, the black-box way of using FNO for fusion simulation can still be prohibitively expensive given that (1) production fusion codes usually have multiple field variables leading to large parameter counts, and (2) production fusion codes are expensive to run, which may limit the amount of training data that can be collected. Therefore, we propose exploiting the algorithm structure of an existing fusion code to sparsify the connectivity of the FNO model and, hence, reduce the training and inference costs.

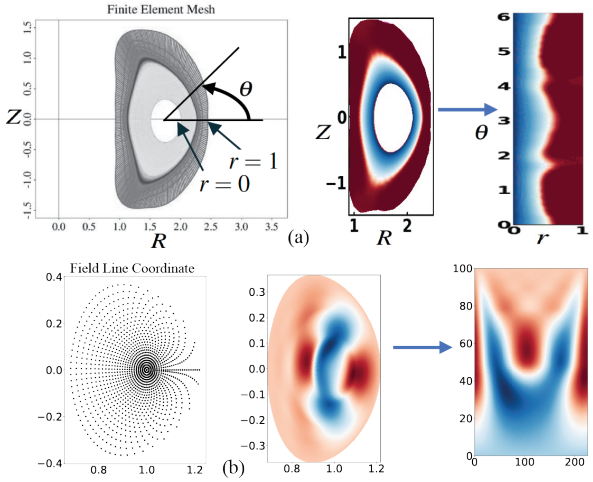

FIG. 2. (a) NIMROD training data: coordinate transformation from the finite-element grid in the $R-Z$ coordinate to the uniform grid in the $r-\theta$ coordinate. (b) GTC training data: data sampled in the radial-poloidal $(\psi-\theta)$ coordinate can be directly viewed as 2D arrays.

B. ST-FNO for NIMROD

1. *Identification of field quantities and their dependencies*

NIMROD[1,33] is an extended-MHD code that uses a mixed spatial discretization of 2D high-order finite-elements and a spectral Fourier decomposition in the 3rd (periodic) direction. For toroidal geometries, the finite-element method (FEM) is applied over the poloidal plane, and Fourier expansion is used for the toroidal direction. The mixed implicit/semi-implicit leapfrog time-advance algorithm in NIMROD exploits the structure of the governing MHD equations. We first discuss the extended-MHD equations and identify the independent field quantities and their dependencies in order to design an efficient ST-FNO model. In what follows, we assume that each field quantity F consists of both equilibrium and perturbed components $F = F_0 + \delta F$, but the ST-FNO model only operates on the perturbed quantities as input and output, and does not use the equilibrium quantities. Also, we use the notation F^k to denote the perturbed field quantity discretized at time step k.

We consider an extended-MHD model with separate electron dynamics and first-order finite-orbit-radius effects that also include both carbon and deuterium ion species. The center-of-mass velocity, $\mathbf{V}$, is computed from the momentum

equation

$$\rho \frac{\partial \mathbf{V}}{\partial t} + \rho(\mathbf{V}\cdot\nabla)\mathbf{V} = \mathbf{J}\times\mathbf{B} - \nabla p - \nabla\cdot\mathbf{\Pi}\,, \qquad (1)$$

where $\rho = m_d n_d + m_c n_c + m_e n_e$ is the total mass density with n_c, n_d and n_e being the number densities for the carbon, deuterium and electron species, respectively. The pressure is determined by the ideal-gas law, $p = k_B(n_e T_e + n_c T_c + n_d T_d)$, with a shared-ion temperature, $T_c = T_d = T_i$. The $\mathbf{\Pi}$ term is the stress tensor, which represents the traceless contributions to the pressure tensor and the exact form includes perpendicular, parallel and gyro-viscosity as described in prior work[34] with straight-forward modifications for multiple ion species. The current density, $\mathbf{J}$, is computed with the pre-Maxwell's Ampere's law, $\mathbf{J} = \nabla\times\mathbf{B}/\mu_0$, as appropriate for low-frequency MHD dynamics. One of the objectives of this work is to leverage the variable dependencies of the equations when constructing the FNO models. However, the center-of-mass velocity equation depends on all of the evolved variables: density, velocity, temperature and magnetic field. That said, our experiments indicate that dropping the dependency on density in our FNO model doesn't significantly affect the model accuracy.

The magnetic field is updated via the induction equation, which combines Faraday's law with a generalized Ohm's law,

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla\times\left[\mathbf{V}\times\mathbf{B} + \frac{1}{n_e e}(\mathbf{J}\times\mathbf{B} - \nabla p_e - \nabla\cdot\mathbf{\Pi}_e) - \eta\mathbf{J}\right] \qquad (2)$$

Thus, the magnetic field is advected by the electron velocity, $\mathbf{v}_e = \mathbf{V} - \mathbf{J}/n_e e$. The resistivity, η, is density and temperature dependent as determined by a Spitzer expression with a multi-ion electron collision time. A neoclassical current drive is applied through $\mathbf{\Pi}_e$[35] where neoclassical poloidal flow damping is also included in the ion stress tensor. The ∇p_e and inverse electron dependencies are present in the induction equation. However, the dominant dependency is on the ideal-MHD center-of-mass velocity advective term, which involves only velocity and magnetic field.

The density equations for only the ion species are evolved and the electron density is determined by the quasineutrality relation, $n_e = n_d + Z_c n_c$. The ion number densities are updated by the continuity equation,

$$\frac{\partial n_\alpha}{\partial t} + \nabla\cdot(n_\alpha\mathbf{V}) = \nabla\cdot D_{n_\alpha}\nabla n_\alpha + \nabla\cdot D_{n_\alpha,hypd}\frac{1}{R^2}\frac{\partial^2}{\partial\phi^2}\nabla n_\alpha, \qquad (3)$$

where α is a species index. The last two terms are numerical density diffusion terms added for computational practicality. The last term, a ϕ-directed hyperdiffusivity, is effective to resolve electro-static turbulence which may become prevalent when using an extended-MHD model[36]. The density has a simple dependence on only itself and the velocity.

Finally, the temperature T_α, $\alpha = i, e$ is updated from the energy equation,

$$\frac{n_\alpha}{\gamma - 1}\left(\frac{\partial T_\alpha}{\partial t} + \mathbf{V}_\alpha\cdot\nabla T_\alpha\right) = -\nabla\cdot\mathbf{q}_\alpha - p_\alpha\nabla\cdot\mathbf{V}_\alpha\,, \qquad (4)$$

where $p_i = k_B(n_c + n_d)T_i$ and $p_e = k_B n_e T_e$ are the ion and electron pressures, respectively. $\mathbf{q}_\alpha$ is the conductive heat flux that includes perpendicular, parallel and cross contributions as described in prior work[34,37] again with straight-forward modifications for multiple ion species. Heating terms are not included in this present NIMROD modeling but in principle can be included in future modeling in a straight-forward manner. The temperature/energy/pressure equations depend on all of the variables, density, velocity, temperature and magnetic field where the magnetic field is important for anisotropic thermal conduction and electron advection. However, we choose to drop the dependency of ion temperature on the magnetic field in our FNO model. Also, we drop the weak dependency on density in our FNO model.

Overall, the system is composed of 10 evolved field quantities: 3 for $\mathbf{V}$, 3 for $\mathbf{B}$, 2 for the ion number density n_c and n_d, and 2 for the temperature, T_e and T_i, and we use their sparse dependencies to design the proposed ST-FNO architecture.

2. *Network design*

In this paper, we focus on a tokamak simulation case where the inner core region of the poloidal plane is not meshed (see Fig. 2(a) for the FEM mesh). The dynamics in this region are not expected to be significant for this particular application. To prepare training data for the FNO models, we first perform a coordinate transformation from the $R-Z$ coordinate to the $r-\theta$ coordinate, where $r = 0$ at the inner boundary and $r = 1$ at the outer boundary (see Fig. 2(a)). We generate 2D arrays for each field quantity in a fixed toroidal plane using a structured grid in the $r-\theta$ coordinate. These arrays are evaluated by evaluating the spectral basis functions of the field data on the FEM mesh. Let F denote one of the 10 field quantities. The training data is labeled $F^k \in \mathbb{R}^{n_r\times n_\theta}$ for each time step k. Note that for this simulation case, there is no dramatic change in the magnitude of each field quantity across time steps, but a significant scale difference exists across different quantities. To address such a scale difference, we normalize each field data F by its mean $\mu_F = \sum_{i,j,k} F^k(i,j)/(n_t n_\theta n_r)$ and standard deviation σ_F as $F \leftarrow (F - \mu_F)/\sigma_F$. In this study, the purpose of the networks is to predict all 10 (unnormalized) field quantities in the $\phi = 0$ poloidal plane at time steps (or snapshots if the fields are collected very few time steps) $k+1,\ldots,k+k_o$ from their history at time steps (or snapshots) $k-k_i+1,\ldots,k$.

Our baseline FNO model is shown in Fig. 1(a) consisting of 1 lifting operator P_{10}, 1 projection operator Q_{10} and 4 FNO layers $F_{10,10}$. The parameter count of the baseline FNO model is dominated by the FNO layers as about $4\times 10^2 M^2 W^2 = 400 M^2 W^2$. In comparison, the proposed ST-FNO model is shown in Fig. 1(e) consisting of lifting and projection operators in 4 groups, as well as 4 FNO layers each having multiple FNO operators. Each FNO operator $F_{m,n}$ corresponds to one governing equation in Section II B 1 and exploits the same structure of the MHD equations that is exploited by the semi-implicit leapfrog algorithm within NIMROD[1]. Due to sparser connectivity, the parameter count can be estimated as $4\times(8\times 3+5\times 2+7\times 1+4\times 1+6\times 3)M^2W^2 = 252M^2W^2$, which is about 37% smaller than the baseline-FNO model.

We note that the typical Mean Squared Errors (MSE) in ma-

chine learning community is not a suitable error metric or loss function for multi-variable fusion simulation data, as it cannot distinguish the magnitude difference between variables. For both the baseline and the ST-FNO models, we propose the following loss function:

$$loss_F = \frac{1}{n_F|\mathbb{K}|}\sum_{k\in\mathbb{K}}\sum_{s=1}^{n_F}\sqrt{\frac{\sum_{i=1}^{k_o}|F_s^{k+i}-\bar{F}_s^{k+i}|_2^2}{\sum_{i=1}^{k_o}|F_s^{k+i}|_2^2}} \tag{5}$$

where $\bar{F}_s$ and F_s denote the sth predicted and ground truth field quantity, and $n_F = 10$. $|F|_2$ denotes the L2 norm of the vectorization of the matrix F. $\mathbb{K}$ denotes the set of training or testing snapshot indices, $|\mathbb{K}| = n_{train}$ for the training error, $|\mathbb{K}| = n_{test}$ for the test error, and $|\mathbb{K}| = 1$ for the error of each individual test index. Unlike many other existing loss functions[24], this formula can faithfully represent the average relative difference between simulation and FNO outputs. Here $\bar{F}_s$ and F_s are the normalized field data, but equation (5) can be used for unnormalized field data as well.

C. ST-FNO for GTC

1. Identification of field quantities and their dependencies

GTC[7] is an advanced particle-in-cell code for simulating plasma turbulence in fusion reactors capable of modeling kinetic electrons, kinetic ions, electromagnetic waves, etc. In this paper, we consider a GTC-generated dataset for simulating the internal kink mode in the low frequency, long-wavelength limit. To develop a ST-FNO architecture leveraging the algorithm structure of GTC, we first summarize the governing equations used to derive the GTC algorithm. To be consistent with the kink mode dataset, we assume that electrons are treated as fluid, and (thermal) ions are treated as gyrokinetic particles[38]. Our ST-FNO design for GTC, just like ST-FNO for NIMROD, only operates on perturbed physical quantities, and we focus on the explanation of the perturbed quantities in what follows.

The perturbed ion particle distribution function δf_i, or equivalently ion particle weight w is updated from the Vlasov equation:

$$\begin{aligned}\frac{dw}{dt} = (1-w)[&-(v_{||}\frac{\delta\mathbf{B}_\perp}{B_0}+\mathbf{v}_E)\cdot\nabla\ln f_{i0} - \frac{\mu v_{||}\hat{\mathbf{b}}_0\cdot\nabla\langle\langle\delta B_{||}\rangle\rangle}{T_i}\\ &+\frac{Z_i}{T_i}(v_{||}E_{||}-\mathbf{v}_d\cdot\nabla(\phi+\frac{\mu}{Z_i}\langle\langle\delta B_{||}\rangle\rangle))].\end{aligned} \tag{6}$$

Here, the perturbed perpendicular magnetic field $\delta\mathbf{B}_\perp$ is computed from the parallel magnetic potential $\delta A_{||}$ as $\delta\mathbf{B}_\perp = \nabla\times\delta A_{||}\hat{\mathbf{b}}_0$. $\mathbf{v}_E = \frac{c\hat{\mathbf{b}}_0\times\nabla\phi}{B_0^*}$ is the E×B drift velocity with ϕ being the electrostatic potential, $\mathbf{v}_d$ is the sum of the magnetic curvature drift current and the magnetic gradient drift current. The parallel particle velocity $v_{||}$ and the gyrocenter $\mathbf{R}$ are updated from the equations of motion. After temporal discretization, one can realize that the distribution function δf_i^{k+1} at time step $k+1$ only depends on δf_i^k, ϕ^k and $A_{||}^k$. Note that δf_i is a particle quantity in GTC, but our ST-FNO model only operates on field quantities. Therefore, we use ion particle density n_i, ion flow velocity u_i, ion perpendicular pressure $\delta P_{\perp i}$ and ion parallel pressure $\delta P_{||i}$ to approximately represent δf_i:

$$n_i = \frac{2\pi B_0^*}{m}\int dv_{||}d\mu\delta f_i \tag{7}$$

$$u_i = \frac{2\pi B_0^*}{m}\int dv_{||}d\mu v_{||}\delta f_i \tag{8}$$

$$\delta P_{\perp i} = \frac{2\pi B_0^*}{m}\int dv_{||}d\mu\mu B_0^*\delta f_i \tag{9}$$

$$\delta P_{||i} = \frac{2\pi B_0^*}{m}\int dv_{||}d\mu m v_{||}^2\delta f_i \tag{10}$$

Although these quantities cannot be directly used in the GTC, ST-FNO can learn to predict these low-order fluid moments of e.g., the kink-mode evolution based on the same fluid moments extracted from GTC results, which effectively provides closure information for the kinetic system.

The perturbed parallel magnetic potential $A_{||}$ is updated via Faraday's law:

$$\frac{\partial A_{||}}{\partial t} = \hat{\mathbf{b}}_0\cdot\nabla(\phi_{eff}-\phi) \tag{11}$$

where the effective electrostatic potential ϕ_{eff} is computed from the perturbed electron number density δn_e and the perturbed parallel magnetic field $\delta B_{||}$ by:

$$\frac{e\phi_{eff}}{T_e} = \frac{\delta n_e}{n_{0e}} + \frac{\delta B_{||}}{B_0} - \frac{\partial\ln n_0}{\partial\psi_0}\delta\psi - \frac{\partial\ln n_0}{\partial\alpha_0}\delta\alpha + \frac{e}{T_e}\frac{\partial\phi_{eq}}{\partial\psi_0}\delta\psi. \tag{12}$$

Here, $\delta\psi$ and $\delta\alpha$ represent the Clebsch representation of $\delta\mathbf{B}_\perp$, and $\delta B_{||}$ can be directly computed from $\delta P_{\perp i}$. Therefore, one can conclude that the parallel magnetic potential at time step $k+1$, $A_{||}^{k+1}$, only depends on $A_{||}^k$, ϕ^k, δn_e^k and $\delta P_{\perp i}^k$.

The perturbed electron number density δn_e can be updated by the electron continuity equation:

$$\begin{aligned}\frac{\partial\delta n_e}{\partial t} = -\nabla\cdot\Bigg[&n_{0e}u_{||e}\left(\frac{\mathbf{B}_0+\delta\mathbf{B}_\perp}{B_0}\right)+n_e\mathbf{v}_E - \frac{P_{||e}\hat{\mathbf{b}}_0\times\boldsymbol{\kappa}}{eB_0}\\ &-\frac{P_{\perp e}\hat{\mathbf{b}}_0\times\nabla B_0}{eB_0^2}-\frac{P_{\perp e}\hat{\mathbf{b}}_0\times\nabla\delta B_{||}}{eB_0^2}\Bigg]\end{aligned} \tag{13}$$

where $n_e = n_{0e}+\delta n_e$, $\boldsymbol{\kappa} = (\hat{\mathbf{b}}_0\cdot\nabla)\hat{\mathbf{b}}_0$ (field line curvature), $P_{||e} = P_{e0}+\delta P_{||e}$, $P_{\perp e} = P_{e0}+\delta P_{\perp e}$. In (13) δu_e is the perturbed electron flow velocity from Ampere's law $\frac{4\pi}{c}en_e u_{||e} = \nabla_\perp^2 A_{||} + \frac{4\pi}{c}Z_i n_i u_{||i}$, $\mathbf{v}_E$ depends on ϕ, the perturbed diamagnetic drift velocity $\mathbf{v}_* = \frac{1}{n_0 m_e\Omega_e}\hat{\mathbf{b}}_0\times\nabla(\delta P_{||e}+\delta P_{\perp e})$ with perturbed pressures $\delta P_{||e}$ and $\delta P_{\perp e}$ depending on ϕ_{eff} and $\delta\mathbf{B}_\perp$. $\delta B_{||}$ depends on $\delta P_{\perp i}$. Recall their dependencies explained above, the perturbed electron number density at time step $k+1$, δn_e^{k+1}, only depends on δn_e^k, $A_{||}^k$, δu_i^k, $\delta P_{\perp i}^k$ and ϕ^k.

In addition to the above-described governing equations and quantity dependencies, the perturbed electrostatic potential at time step $k+1$, ϕ^{k+1}, can be computed from δn_e^{k+1} and δn_i^{k+1}

via the Poisson equation:

$$\frac{Z_i^2\rho_i^2}{T_i}\sum_{s\neq e}\frac{n_{0s}m_s}{m_i}\nabla_\perp^2\phi = -(1-\rho_i^2\nabla_\perp^2)\sum_s Z_s\bar{n}_s \qquad (14)$$

Overall, we choose 7 independent field quantities δn_e, δn_i, δu_i, $\delta P_{\perp i}$, $\delta P_{||i}$, $A_{||}$ and ϕ, and use their sparse dependencies to design the proposed ST-FNO architecture.

2. Network design

In principle, we can apply the same methodology as the case of NIMROD to design ST-FNO for GTC. Similar to the ST-FNO for the NIMROD case, here we aim at predicting all 7 field quantities at time steps (or snapshots) $k+1,\ldots,k+k_o$ from their history at time steps (snapshots) $k-k_i+1,\ldots,k$.

However, there are three significant differences. First, the NIMROD extended-MHD simulations considered here only consider fluid dynamics, but GTC is a PIC code that evolves both field and particle data. As explained in Section II C 1, we can use n_i, u_i, $\delta P_{\perp i}$ and $\delta P_{||i}$ to describe the lowest-order moments of the particle distribution data δf_i. Second, unlike NIMROD whose simulation data in poloidal planes is represented with FEM mesh, GTC's simulation data in poloidal planes is directly sampled on the field-aligned coordinate with radial flux coordinate ψ and poloidal coordinate θ, leading to 2D arrays (see Fig. 1(b)) as the training data. Let $F = \delta n_e, \delta n_i, \delta u_i, \delta P_{\perp i}, \delta P_{||i}, A_{||}, \phi$ be one of the 7 field quantities for ST-FNO, $F \in \mathbb{R}^{n_\psi \times n_\theta}$. In principle, a coordinate transformation to a field-aligned coordinate could be performed with the NIMROD data but was not considered in this work. Third, our GTC dataset is generated from a kink mode simulation case, which consists of the linear and nonlinear phases. In the linear phase, several field-quantity magnitudes grow exponentially as time increases while the field patterns remain stable; in contrast, in the nonlinear phase, the field magnitudes stay stable, but the field patterns can evolve dramatically. The following design addresses this last challenge efficiently.

To design an efficient network surrogate for the kink mode simulation, we normalize each field quantity at time step k, F^k, by its root mean square (RMS) value $[F^k]$ as $F^k \leftarrow F^k/[F^k]$ and use the normalized data to train the ST-FNO model, as depicted in Fig. 1(g). Compared to the baseline FNO model in Fig. 1(b), ST-FNO's sparse connectivity can significantly reduce the network parameter count. To preserve the RMS information for both ST-FNO and baseline FNO, we create a separate network consisting of two fully-connected (FC) layers called RMS-FC model. When $k_i = k_o = 1$, RMS-FC predicts $\log[F^{k+1}]$ from $\log[F^k]$ as shown in Fig. 1(f). We use the following loss function for RMS-FC:

$$loss_{RMS} = \frac{1}{n_F}\sum_{k\in\mathbb{K}}\sum_{i=1}^{k_o}\sqrt{\frac{\sum_{s=1}^{n_F}(\log[F_s^{k+i}]-\log[\bar{F}_s^{k+i}])^2}{\sum_{s=1}^{n_F}(\log[F_s^{k+i}])^2}} \qquad (15)$$

with $\mathbb{K}$ the same as that in (5).

Our baseline FNO model is shown in Fig. 1(b) consisting of 1 lifting operator P_7, 1 projection operator Q_7 and 4 FNO layers $F_{7,7}$. The parameter count of the baseline FNO model is dominated by the FNO layers as about $4\times 7^2M^2W^2 = 196M^2W^2$. In comparison, the proposed ST-FNO model is shown in Fig. 1(g) consisting of 7 lifting and 7 projection operators, as well as 4 FNO layers each having 7 FNO operators. Each FNO operator $F_{m,n}$ corresponds to one governing equation in Section II C 1, except that those for n_i, u_i, $\delta P_{\perp i}$ and $\delta P_{||i}$ approximately representing the Vlasov equation (6) (the connectivity is shown in red). Due to sparser connectivity, the parameter count of ST-FNO can be estimated as $4\times(5+3+3+3+3+4+2)M^2W^2 = 92M^2W^2$, which is significantly smaller than the baseline-FNO model.

TABLE I. Comparison of accuracy and parameter counts of the ST-FNO and the baseline FNO models for the NIMROD case (random splitting). k_i represents the number of past snapshots used as the model input, and k_o represents the number of future snapshots as the module output.

Model	k_i	k_o	Train Error (10^{-3})	Test Error (10^{-3})	Parameter Count (M)
baseline-FNO	1	1	8.76	9.19	92.8
baseline-FNO	3	1	5.61	5.8	92.8
baseline-FNO	5	1	5.65	6.94	92.8
ST-FNO	1	1	7.97	8.7	58.2
ST-FNO	3	1	5.79	6.09	58.2
ST-FNO	5	1	6.13	7.39	58.2

For both the baseline and the ST-FNO models, we use the loss function in (5) with $n_F = 7$. Recall that F_s^k in (5) represent the normalized perturbed field quantities.

D. A Note on Generalizing ST-FNO to Other Codes

Although in this paper we only demonstrate how to design ST-FNO for two simulation codes, NIMROD and GTC, it shall be clear by now that ST-FNO can be easily adapted to other fusion codes and, more generally, codes solving systems of time-dependent PDEs. Assume that we have a set of n_F field variables $\mathbf{F}(x,y) = (F_1,\ldots,F_{n_F})$ satisfying

$$\frac{\partial\mathbf{F}}{\partial t} = \mathscr{L}(\mathbf{F},\nabla\mathbf{F},\nabla^2\mathbf{F},\ldots) \qquad (16)$$

where $\mathscr{L}$ is a vector of differential operators involving only spatial derivatives.

One can identify the dependency of F_s based on the right hand side of (16) and naturally design a FNO model whose connectivity respects such dependency. The training data can be generated using a regular grid in (x,y), regardless of the numerical algorithms being used, e.g., FEM, finite-difference, finite-volume, spectral method, etc. If particle-based methods e.g., PIC, are used, one can convert the particle data to field data, just like (7)-(10) for GTC. In addition, if the field-dependency is weak it can be neglected in the FNO connectivity model. Introducing further sparsity into the NIMROD ST-FNO model may be possible.

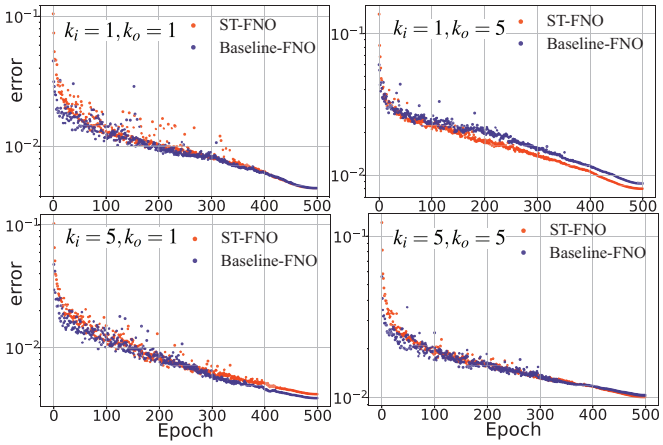

FIG. 3. Test error using (5) vs. epoch number for ST-FNO and baseline FNO for the NIMROD case with $k_i = 1,5$ and $k_o = 1,5$.

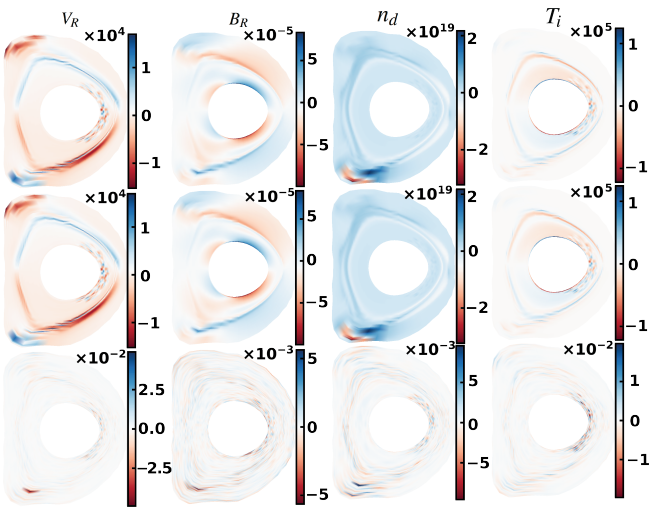

FIG. 4. Color contour plots of perturbed fields from (top) NIMROD simulation results, (middle) ST-FNO results and (bottom) their relative difference using (17) (random splitting) at time step 58400 (snapshot $k = 233$) for 4 out of the total 10 field quantities: V_R, B_R, n_d and T_i. $k_i = 1$, $k_o = 1$.

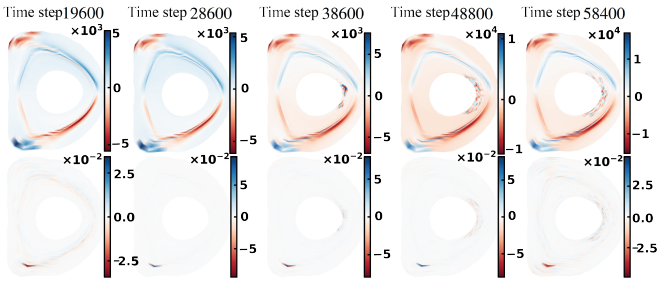

FIG. 5. Color contour plots of perturbed fields from the (top) ST-FNO results and (bottom) their relative difference using (17) (random splitting) with respect to the NIMROD simulation results at time steps $19600, 28600, 38600, 48800, 58400$ for the field quantity V_R. $k_i = 1$, $k_o = 1$.

III. NUMERICAL RESULTS

In this section, we provide numerical results using two distinct fusion simulations, an extended-MHD result from NIMROD and a gyrokinetic PIC result from GTC, to demonstrate the efficiency and accuracy of the proposed ST-FNO methodology.

A. NIMROD results

The data set is generated by a nonlinear simulation using the model described in Sec. II B 1 using the NIMROD code. This case extends prior work[39,40] that studies the quiescent H-mode regime in the DIII-D tokamak. The 3D tokamak-plasma volume is discretized with a 96×256 grid of bi-quartic elements over the poloidal plane and with 22 toroidal Fourier modes over the toroidal angle. For the FNO application, we use a set of 257 snapshots from the NIMROD simulation steps numbered 12000 to 63200 with one FNO snapshot per 200 NIMROD steps. The NIMROD time steps are of size $\Delta t_m = 5 \times 10^{-9}$ s, and at this time-step size and resolution, the compressional Alfvén wave Courant–Friedrichs–Lewy (CFL) condition is approximately 800 while the flow-speed CFL is approximately two. Eleven Perlmutter CPU nodes are used, which requires approximately 7 seconds per step. On a fixed poloidal plane at $\phi = 0$, we generate 2D data with $n_r = 64$ and $n_\theta = 64$ as data for the ST-FNO model.

For both the baseline-FNO and ST-FNO models, we set the interior widths $W = 20$, the number of Fourier modes in each of r and θ directions $M = 12$, and the number of FNO layers to 4. Recall that n_{train} and n_{test} are the number of snapshots in the training and test sets, respectively. Here, we consider two ways of splitting the time steps. (1) Random splitting: the snapshots are randomly split into the training set with $n_{train} = 128$ and the test set with $n_{test} = 129$. (2) Sequential splitting: the snapshots are split into two halves, the first n_{train} snapshots as the training set and the rest as the test set. In other words, we only train the models from data in the past and test the models with data in the future. We train both models for 500 epochs, *i.e.*, 500 network training passes through the data, with the training and testing errors defined by (5). The training is performed on 1 NVIDIA A100 GPU of Perlmutter.

1. *Random splitting*

We first compare the convergence of the test error as a function of the epoch number with $k_i = 1,5$ and $k_o = 1,5$ in Fig. 3. We remark that the convergence of the baseline-FNO and ST-FNO models are very similar. The training and test errors at epoch 500 listed in Table I confirm that the two models have very similar inference accuracy.

Next, we remark that ST-FNO with a parameter count of 67.5 million is more efficient compared to baseline-FNO with a parameter count of 92.8 million, leading to a 30% model size reduction (see Table I). It's worth noting that the FNO parameter count is insensitive to k_i and k_o as the FNO operators dom-

inate the parameter count instead of the lifting/projection operators. For both models, the training time per epoch is about 3-10s and the inference time per time step is about 5-20ms using 1 Perlmutter A100 GPU. The memory requirement for ST-FNO is about 280 MB. In comparison, the approximate solutions in NIMROD are computed with the SuperLU_DIST[41] direct solver used to precondition the algebraic solves, which requires 34 GB memory, 9s factorization time, and 340 ms apply time (summing up the numbers for all physical quantities) for each NIMROD toroidal Fourier mode (or equivalently, for each poloidal plane) using 64 Perlmutter CPU cores. Assuming that the LU factorization and the ST-FNO model can be reused for multiple time steps and their computational cost can be amortized, we can just compare the memory requirement and inference/apply time. To this end, ST-FNO is capable of achieving 120x memory reduction compared with SuperLU_DIST-based solution in predicting an estimate of the field quantities over a single poloidal plane. If one wants to use ST-FNO to provide an initial guess for the algebraic solve at every time step, that requires only $20ms \times 44 \approx 1s$. Compared with the $340ms \times 22 \approx 7.5s$ of SuperLU_DIST-based preconditioner per Krylov iteration, the ST-FNO-based initial guess is computationally inexpensive. Note that the factor of 44 indicates that prediction from 44 poloidal planes are required to resolve the 22 Fourier modes used in NIMROD. If one wants to get an estimate of the fields every 200 time steps e.g., to perform long-term diagnostics/prediction, ST-FNO only takes $1s$. In contrast, the first-principle numerical solver takes $200 \times 7 \approx 1400s$.

Next, we compare the field plots predicted by ST-FNO and simulated by NIMROD. Fig. 4 plots the NIMROD simulation results, ST-FNO results, and their relative difference for 4 out of the 10 field quantities at time step 58400. For a NIMROD field quantity F^k and its ST-FNO output $\bar{F}^k$, the relative difference is computed as:

$$relative\ diff = (F^k - \bar{F}^k)/\max_{i,j}|F^k(i,j)| \qquad (17)$$

Note that ST-FNO can attain $10^{-2} - 10^{-3}$ relative difference for most field quantities. Furthermore, Fig. 5 shows the ST-FNO results for V_R and the relative difference with respect to the NIMROD results for 5 different time steps (from the test set), a relative difference of 10^{-2} can be observed. Notable dynamics are developed near the edge of the inner core, and the largest error appears near the magnetic x-point. This point is a stagnation point of the poloidal magnetic field where field-lines on the last-closed magnetic-flux surface will asymptotically approach the x-point. This makes this spatial location unique within the simulated domain and thus exhibits differing dynamics relative to the rest of the domain.

Next, we examine the inference error per test sample in the random splitting setting. Fig. 6(a) plots the relative error of the predicted data $\bar{F}_s^k = M_{FNO}(F_s^{k-1})$ for each sample snapshot k. Here $y = M_{FNO}(x)$ denotes the ST-FNO model with input x and output y. In other words, this represents a single forward pass of the input x through the ST-FNO model. As the reference, we also plot the relative error for directly using F_s^{k-1} as an estimate for F_s^k (dubbed "previous iterate (PI)"), i.e., $\bar{F}_s^k = F_s^{k-1}$. As we can see, ST-FNO inference can outperform PI by up to 5x more accuracy for most test snapshots.

2. *Sequential splitting*

In this subsection, we examine the inference error per test sample in the sequential splitting settings (see Fig. 6(b)), we set n_{train} to different values and try to investigate how many more snapshots ST-FNO can predict a valuable solution without retraining the model with newer snapshots. First, we consider $k_i = 1$ and $k_o = 1$. The FNO prediction is $\bar{F}_s^k = M_{FNO}(F_s^{k-1})$ with $n_{train} = 130, 180, 220, 240$. In other words, we always predict the next snapshot $\bar{F}_s^k$, using the model M_{FNO}, with the NIMROD simulation data at previous snapshot F_s^{k-1} as the input. Note that for each curve, the ST-FNO model is trained only once. In comparison, we also plot results using PI $\bar{F}_s^k = F_s^{k-1}$. We remark that the ST-FNO prediction can always outperform PI for the first 3-4 snapshots, then a retraining of the model becomes necessary. Note that the PI estimate quality drops significantly for snapshot number $k > 250$, which justifies the need for ST-FNO prediction (with more frequent retraining).

In addition, we visualize the fields predicted by ST-FNO (in the sequential splitting settings with $k_i = 1$ and $k_o = 1$) and simulated by NIMROD. Fig. 7 plots the NIMROD simulation results, ST-FNO results, and their relative difference defined by (17) for the field quantity V_R at two time steps. For time step 48000 (snapshot 181), we consider with ST-FNO model with $n_{train} = 130$ and $n_{train} = 180$. For time step 56000 (snapshot 131), we consider the ST-FNO model with $n_{train} = 130$ and $n_{train} = 180$. It is clear from both Fig. 7 and Fig. 6(b) that retraining the ST-FNO model with more available snapshot data can significantly improve the model prediction accuracy.

Next, we consider $k_i = 5$ and $k_o = 5$. In other words, we try to predict the field at a time scale 5 times larger than the above experiment. The ST-FNO prediction is $\bar{F}_s^{k,\ldots,k+4} = M_{FNO}(F_s^{k-5,\ldots,k-1})$ with $n_{train} = 130, 180, 220, 240$ (see Fig. 6(c)). In comparison, we also plot results using PI $\bar{F}_s^k = F_s^{k-5}$. Note for snapshot $k+4$, ST-FNO uses all 5 snapshots $k-5, \ldots, k-1$, while PI uses only snapshot $k-1$. Therefore, ST-FNO can significantly outperform PI for almost all the data points with $n_{train} = 180, 220, 240$.

Finally, we consider using ST-FNO in an auto-regressive fashion. In other words, we want to predict the field $\bar{F}_s^k$ from input quantities that are k_0 snapshots back in time: $\bar{F}_s^k = M_{FNO}(\bar{F}_s^{k-1}), \bar{F}_s^{k-1} = M_{FNO}(\bar{F}_s^{k-2}), \ldots, \bar{F}_s^{k-k_0+1} = M_{FNO}(F_s^{k-k_0})$. We trained the ST-FNO model with $k_i = k_o = 1$ using $n_{train} = 130$ snapshots, and applied the trained model with k_o varying from 1 to 5. First, the error increases over time for all curves, just like the other subfigures, in Fig. 6(d). Note that the "$k_0 = 1$" curve is the same as the "$n_{train} = 130$" curve in Fig. 6(b). Next, one can clearly see that increasing k_0 will quickly increase the prediction error due to error accumulation of autoregression. This suggests that both a proper fine-tuning/retraining of the ST-FNO model over time and a proper choice of number of autoregression steps are

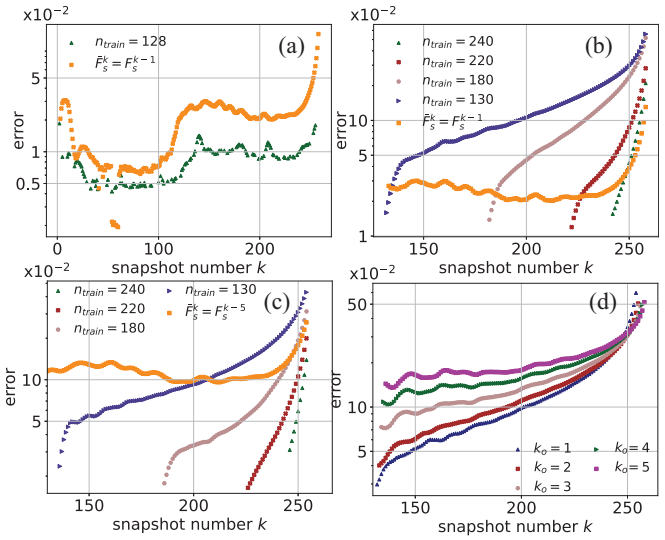

FIG. 6. Relative error using (5) of the estimated fields $\bar{F}_s^k$ for each test sample in the NIMROD case. Each snapshot k represents 200 time steps in the NIMROD code, and snapshot 1 represents time step 12000. (a) Random splitting setting: $\bar{F}_s^k$ is computed from ST-FNO $\bar{F}_s^k = M_{FNO}(F_s^{k-1})$ with $n_{train} = 128$ and the previous iterate (PI) $\bar{F}_s^k = F_s^{k-1}$. $k_i = k_o = 1$. (b) Sequential splitting setting: $\bar{F}_s^k$ is computed from ST-FNO $\bar{F}_s^k = M_{FNO}(F_s^{k-1})$ with $n_{train} = 130, 180, 220, 240$ and PI $\bar{F}_s^k = F_s^{k-1}$. $k_i = k_o = 1$. (c) Sequential splitting setting: Same as (b) but with $k_i = k_o = 5$. In other words, $\bar{F}_s^{k,\ldots,k+4} = M_{FNO}(F_s^{k-5,\ldots,k-1})$. (d) Sequential splitting setting: The ST-FNO operates in an auto-regressive fashion to predict snapshot k from $F_s^{k-k_0}$: $\bar{F}_s^k = M_{FNO}(\bar{F}_s^{k-1}), \bar{F}_s^{k-1} = M_{FNO}(\bar{F}_s^{k-2}), \ldots, \bar{F}_s^{k-k_0+1} = M_{FNO}(F_s^{k-k_0})$.

needed.

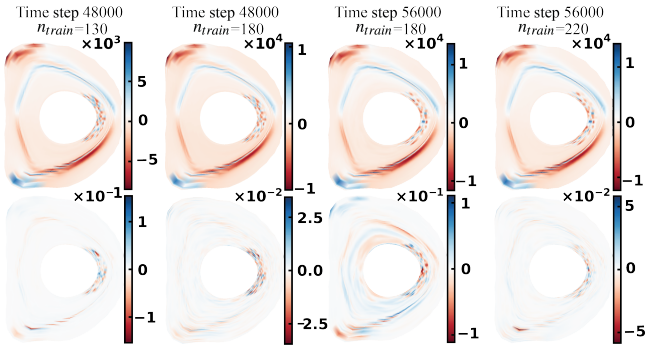

FIG. 7. Color contour plots of perturbed fields from (top) ST-FNO results and (bottom) their relative difference using (17) with respect to the NIMROD simulation results (sequential splitting) at time steps 48000, 56000 with different sized training data for the field quantity V_R. $k_i = 1$, $k_o = 1$.

B. GTC results

The data set is generated by a nonlinear gyrokinetic simulation using GTC for the DIII-D discharge #141216 at $t = 1750$

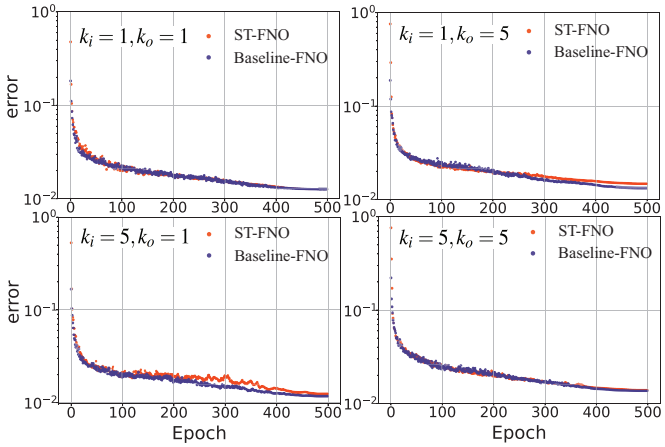

FIG. 8. Test error using (5) for the normalized field data vs. epoch number for ST-FNO and baseline FNO for the GTC case with $k_i = 1, 5$ and $k_o = 1, 5$.

TABLE II. Comparison of accuracy and parameter counts of the ST-FNO and the baseline FNO models for the GTC case (random splitting). k_i represents the number of past snapshots used as the model input, and k_o represents the number of future snapshots as the module output.

Model	k_i	k_o	Train Error (10^{-3})	Test Error (10^{-2})	Parameter Count (M)
baseline-FNO	1	1	9.96	1.78	45.4
baseline-FNO	3	1	9.51	1.77	45.4
baseline-FNO	5	1	8.23	1.79	45.4
ST-FNO	1	1	10.8	1.77	21.2
ST-FNO	3	1	7.93	1.81	21.2
ST-FNO	5	1	7.95	1.85	21.2

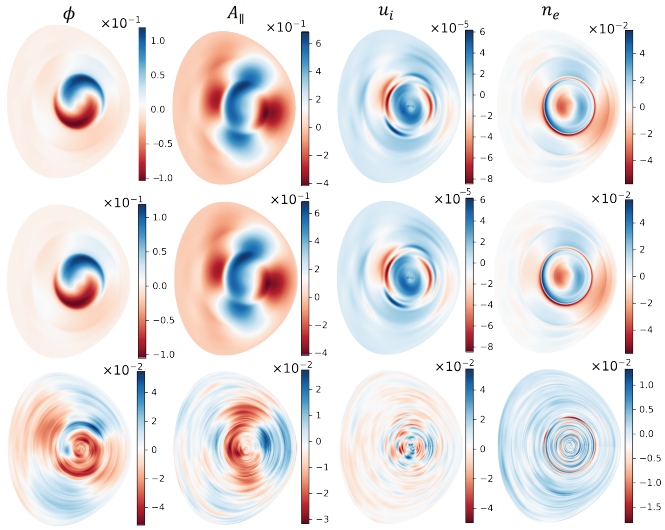

FIG. 9. Color contour plots of perturbed fields from (top) GTC simulation results, (middle) ST-FNO results and (bottom) their relative difference using (17) (random splitting) at time step 16000 for 4 out of the total 7 field quantities: $\phi, A_{||}, u_i$ and n_e. $k_i = 1$, $k_o = 1$.

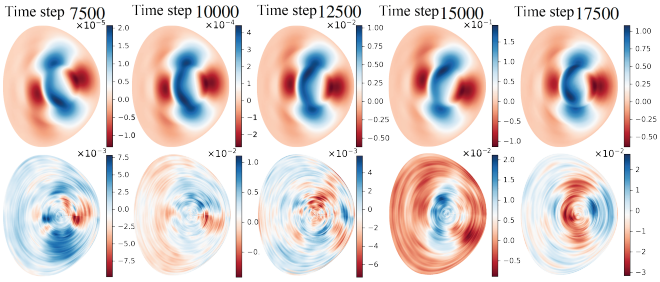

FIG. 10. Color contour plots of perturbed fields from the (top) ST-FNO results and (bottom) their relative difference using (17) with respect to the GTC simulation results (random splitting) at time steps $7500, 10000, 12500, 15000, 17500$ for the field quantity $A_{||}$. $k_i = 1$, $k_o = 1$.

ms. The DIII-D tokamak geometry is discretized with a $100 \times 250 \times 24$ mesh in radial, poloidal, and parallel directions; the time step size is set to $\Delta t = 1.483 \times 10^{-8} s$. We run the simulation for 20000 time steps and keep both $n = 0$ and $n = 1$ modes in the simulation. This simulation requires 3.9 hours on 6 Perlmutter GPU nodes. The simulation generates one snapshot per 100 time steps, and we use time steps 4000 to 18800 as the dataset for training and testing ST-FNO. This is due to the fact that the first 3999 time steps consist of significant initial noises, and the last 200 time steps become physically unreliable due to numerical instabilities. Note that there is a transition point near time step 15300 (snapshot number $k = 114$) that separates the linear phase and nonlinear kink mode phase.

For both the baseline-FNO and ST-FNO models, we set the interior widths $W = 20$, the number of Fourier modes in each direction $M = 12$, and the number of FNO layers to 4. Here, we consider two ways of splitting the time steps. (1) Random splitting: the snapshots are randomly split into the training set with $n_{train} = 74$ and the test set with $n_{test} = 75$. (2) Sequential splitting: the snapshots are split into two halves, the first n_{train} snapshots as the training set and the rest as the test set. In other words, we only train the models from data in the past and test the models with data in the future. We train the models for 500 epochs with the errors defined by (5) for the ST-FNO and baseline-FNO model and (15) for the RMS-FC model. The training and testing are performed with 1 NVIDIA A100 GPU of Perlmutter.

1. *Random splitting*

We first compare the convergence of the test error in the random splitting setting as a function of the epoch number with $k_i = 1, 5$ and $k_o = 1, 5$ in Fig. 8. We remark that the convergence of the baseline-FNO and ST-FNO models are very similar. The training and test errors at epoch 500 listed in Table II confirm that the two models have very similar inference accuracy. Next, we remark that ST-FNO with a parameter count of 21.2 million is more efficient compared to baseline-FNO with a parameter count of 45.4 million, leading to a 2X

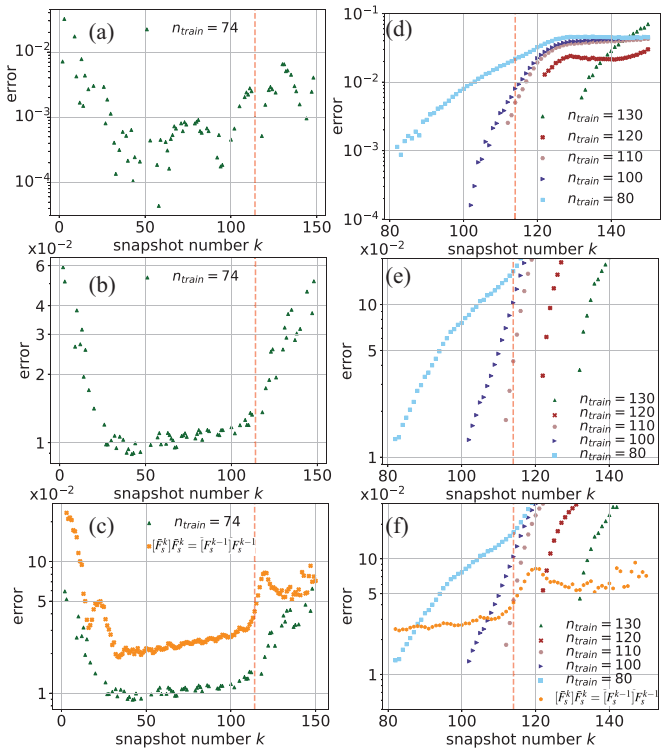

FIG. 11. Relative error using (5) of the estimated fields $\bar{F}_s^k$ for each test sample in the GTC case with $k_i = k_o = 1$. Each snapshot k represents 100 time steps in the GTC code, and snapshot 1 represents time step 4000. The vertical dashed line represents the linear-to-nonlinear transition point at time step 15300 (i.e., snapshot $k = 114$). Left (random splitting setting): $n_{train} = 74$ and the previous iterate (PI) $[\bar{F}_s^k]\bar{F}_s^k = [F_s^{k-1}]F_s^{k-1}$. Right (sequential splitting setting): $n_{train} = 80, 100, 110, 120, 130$ and PI results. Top: Relative error of the RMS values using (15) for (a) random splitting and (d) sequential splitting. The quantities are computed as $[\bar{F}_s^k] = M_{RMS}([F_s^{k-1}])$ with M_{RMS} denoting the RMS-FC model. Middle: Relative error using (5) for normalized fields for (b) random splitting and (e) sequential splitting. The quantities are computed as $\bar{F}_s^k = M_{FNO}(F_s^{k-1})$ with M_{FNO} denoting the ST-FNO model. Bottom: Relative error using (5) for fields de-normalized with the RMS data as $[\bar{F}_s^k]\bar{F}_s^k$ for (c) random splitting and (f) sequential splitting.

model size reduction (see Table II). For ST-FNO models, the training time per epoch is about 6.78 s and the inference time per snapshot (i.e., per 100 time steps) is about 2.00 ms. In comparison, the average time per time step in GTC is about 0.7s.

Next, we compare the field plots predicted by ST-FNO and simulated by GTC in the random splitting setting. Fig. 9 plots the GTC simulation results, ST-FNO results, and their relative difference for 4 field quantities at time step 16000. Note that these plots show the combination of the ST-FNO results and the RMS-FC results. Note that ST-FNO can attain $10^{-2} - 10^{-3}$ relative difference for all field quantities. Furthermore, Fig. 10 shows the ST-FNO results for $A_{||}$ and the relative difference with respect to the GTC results for 5 different time steps (from the test set), a relative difference of $10^{-2} - 10^{-3}$ can be observed. Note that these time steps cover

both the linear and nonlinear phases. The RMS values range from $10^{-5}-10^{0}$ and the kink mode exhibits dramatic pattern changes, but our model can still predict very well.

Next, we examine the inference error per test sample in the random splitting settings. Fig. 11(left) plots the relative error of the normalized data, RMS, and the de-normalized data. Roughly speaking, let $[\bar{F}]=[F](1+\varepsilon_{RMS})$ be the model-generated RMS data and $\bar{F}=F(1+\varepsilon_{norm})$ be the model-generated normalized field data. In the random splitting setting (left column), the RMS values can be very computed as $[\bar{F}_s^k]=M_{RMS}([F_s^{k-1}])$ with M_{RMS} denoting the RMS-FC model. The inference errors are $\varepsilon_{RMS}\approx 10^{-4}-10^{-2}$ given the presence of both linear and nonlinear phase data in the training set (see Fig. 11(a)). Similarly, the normalized field data can be accurately computed as $\bar{F}_s^k=M_{FNO}(F_s^{k-1})$ with at least $|\varepsilon_{norm}|\approx 6\times 10^{-2}$ error (see Fig. 11(b)). The de-normalized field $[\bar{F}]\bar{F}\approx[F]F(1+\varepsilon_{RMS}+|\varepsilon_{norm}|)$ shows a inference error about 6×10^{-2} (see Fig. 11(c)). Note that for both the RMS and the field data, the inference error is larger towards snapshots 4000 and 18800 due to the presence of more out-of-distribution data in these regions. In comparison, we also show the PI results for the de-normalized fields: $[\bar{F}_s^k]\bar{F}_s^k=[F_s^{k-1}]F_s^{k-1}$ in Fig. 11(c). The proposed ST-FNO+RMS-FC model can yield up to 5x better inference accuracy compared with PI, but in the nonlinear phase their difference becomes less significant.

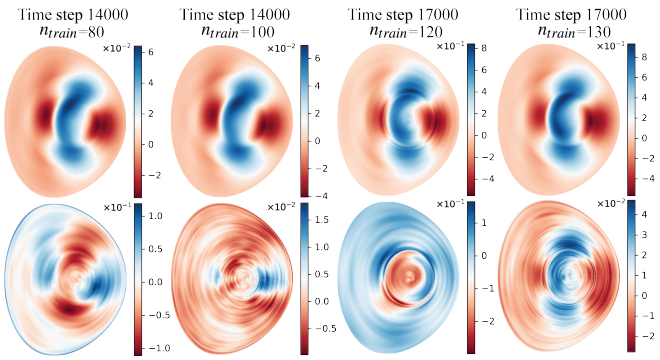

FIG. 12. Color contour plots of perturbed fields from (top) ST-FNO results and (bottom) their relative difference using (17) with respect to the GTC simulation results (sequential splitting) at time steps 14000, 17000 with different sized training data for the field quantity $A_{||}$. $k_i=1$, $k_o=1$.

2. *Sequential splitting*

Finally, we examine the prediction error per test sample in the sequential splitting settings (see Fig. 11(right column)). We set n_{train} to different values and try to investigate how many more snapshots ST-FNO can predict a valuable solution (e.g. with the error below approximately 0.1) without retraining the model with newer snapshots. As expected, the errors go up as the snapshot index increases. That said, for RMS (Fig. 11(d)), one can accurately predict the results in the linear phase for the next 30-40 snapshots with only $n_{train}=80$ training samples. For the normalized data (Fig. 11(e)) and denormalized data (Fig. 11(f)), the linear phase can be accurately predicted with $n_{train}=80,100,110$. However, in the nonlinear phase, the ST-FNO can only predict the next snapshot more accurately than PI, suggesting that retraining is needed for every snapshot.

In addition, we visualize the fields predicted by ST-FNO (in the sequential splitting settings with $k_i=1$ and $k_o=1$) and simulated by GTC. Fig. 12 plots the GTC simulation results, ST-FNO results, and their relative difference defined by (17) for the field quantity $A_{||}$ at two time steps. For time step 14000 (snapshot 101), we consider the ST-FNO model with $n_{train}=80$ and $n_{train}=101$. For time step 17000 (snapshot 131), we consider the ST-FNO model with $n_{train}=120$ and $n_{train}=130$. It is clear from both Fig. 12 and Fig. 11(f) that retraining the ST-FNO model becomes necessary in the sequential splitting setting.

CONCLUSION

This paper proposed an efficient and accurate SciML model, called ST-FNO, which leverages the sparse connectivity indicated by the governing equations of fusion codes. ST-FNO has been applied to an extended-MHD code NIMROD and a gyrokinetic PIC code GTC to demonstrate its inference accuracy, memory efficiency and CPU efficiency. We remark that for fusion simulation codes, or more generally multi-variable time-dependent PDEs with multiple variables, explicitly exploiting the sparsity dependency indicated by the governing equations can effectively reduce the sizes of SciML models without sacrificing the interference accuracy, which is particularly useful for data-scarce scientific applications.

The limitation of ST-FNO is the expensive training cost (just like most other ML models), and we plan to explore the idea of fine-tuning[32] as well as further increasing the number of time steps per snapshot to reduce such cost. It is also likely that one can further sparsify ST-FNO by dropping weaker dependencies using more domain knowledge. A more systematic study of the prediction confidence, particularly for a large number of output time steps, is highly desirable for ST-FNO to be used as a reliable diagnostic tool for critical plasma events. Future work also includes integrating ST-FNO into fusion codes as an ML-based initial high-accuracy guess for preconditioners. Taking NIMROD for an example, this would require (1) conversion of the model output from uniform mesh in $r-\theta$ coordinate to the FEM mesh, (2) extension of the ST-FNO model to predict fully 3D data in the complex Fourier representation used by NIMROD, and (3) an automatic mechanism to determine when to retrain or fine-tune the ST-FNO model on-the-fly.

ACKNOWLEDGMENTS

This work was supported by the Laboratory Directed Research and Development Program of Lawrence Berkeley National Laboratory under U.S. Department of Energy Contract

No. DE-AC02-05CH11231. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231. The contributions from Dr. King are supported by the U.S. Department of Energy under award number DE-AC02-09CH11466.

DATA AVAILABILITY STATEMENT

The machine learning code supporting the findings of this study is available in the GitHub repository at `https://github.com/liuyangzhuan/STFNO`. A selected dataset is available at `https://zenodo.org/records/13901806`. The other datasets are available from the corresponding author upon reasonable request.

[1] C. R. Sovinec, A. Glasser, T. Gianakon, D. Barnes, R. Nebel, S. Kruger, D. Schnack, S. Plimpton, A. Tarditi, M. Chu, *et al.*, "Nonlinear magnetohydrodynamics simulation using high-order finite elements," Journal of Computational Physics **195**, 355–386 (2004).

[2] S. C. Jardin, N. Ferraro, X. Luo, J. Chen, J. Breslau, K. E. Jansen, and M. S. Shephard, "The m3d-c1 approach to simulating 3d 2-fluid magnetohydrodynamics in magnetic fusion experiments," Journal of Physics: Conference Series **125**, 012044 (2008).

[3] G. Huysmans and O. Czarny, "Mhd stability in x-point geometry: simulation of elms," Nuclear Fusion **47**, 659 (2007).

[4] M. Hoelzl, G. Huijsmans, S. Pamela, M. Bécoulet, E. Nardon, F. Artola, B. Nkonga, C. Atanasiu, V. Bandaru, A. Bhole, D. Bonfiglio, A. Cathey, O. Czarny, A. Dvornova, T. Fehér, A. Fil, E. Franck, S. Futatani, M. Gruca, H. Guillard, J. Haverkort, I. Holod, D. Hu, S. Kim, S. Korving, L. Kos, I. Krebs, L. Kripner, G. Latu, F. Liu, P. Merkel, D. Meshcheriakov, V. Mitterauer, S. Mochalskyy, J. Morales, R. Nies, N. Nikulsin, F. Orain, J. Pratt, R. Ramasamy, P. Ramet, C. Reux, K. Särkimäki, N. Schwarz, P. S. Verma, S. Smith, C. Sommariva, E. Strumberger, D. van Vugt, M. Verbeek, E. Westerhof, F. Wieschollek, and J. Zielinski, "The jorek non-linear extended mhd code and applications to large-scale instabilities and their control in magnetically confined fusion plasmas," Nuclear Fusion **61**, 065001 (2021).

[5] T. Goerler, X. Lapillonne, S. Brunner, T. Dannert, F. Jenko, F. Merz, and D. Told, "The global version of the gyrokinetic turbulence code gene," Journal of Computational Physics **230**, 7053–7071 (2011).

[6] J. Candy and R. Waltz, "An eulerian gyrokinetic-maxwell solver," Journal of Computational Physics **186**, 545–581 (2003).

[7] Z. Lin, T. S. Hahm, W. Lee, W. M. Tang, and R. B. White, "Turbulent transport reduction by zonal flows: Massively parallel simulations," Science **281**, 1835–1837 (1998).

[8] J. Li, C. Xiao, Z. Lin, D. Liu, X. Ji, and X. Wang, "GTC simulation of linear stability of tearing mode and a model magnetic island stabilization by ECCD in toroidal plasma," Physics of Plasmas **27** (2020).

[9] S. Ku, C. Chang, and P. Diamond, "Full-f gyrokinetic particle simulation of centrally heated global itg turbulence from magnetic axis to edge pedestal top in a realistic tokamak geometry," Nuclear Fusion **49**, 115021 (2009).

[10] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," Journal of Computational physics **378**, 686–707 (2019).

[11] L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via deeponet based on the universal approximation theorem of operators," Nature Machine Intelligence **3**, 218–229 (2021).

[12] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," arXiv preprint arXiv:2010.08895 (2020).

[13] Y. Cho, J. W. Demmel, J. King, X. S. Li, Y. Liu, and H. Luo, "Harnessing the crowd for autotuning high-performance computing applications," in *2023 IEEE International Parallel and Distributed Processing Symposium (IPDPS)* (IEEE, 2023) pp. 635–645.

[14] Y. Liu, W. M. Sid-Lakhdar, O. Marques, X. Zhu, C. Meng, J. W. Demmel, and X. S. Li, "Gptune: Multitask learning for autotuning exascale applications," in *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (2021) pp. 234–246.

[15] X. He, Y. Choi, W. D. Fries, J. L. Belof, and J.-S. Chen, "glasdi: Parametric physics-informed greedy latent space dynamics identification," Journal of Computational Physics **489**, 112267 (2023).

[16] C. Bonneville, X. He, A. Tran, J. S. Park, W. Fries, D. A. Messenger, S. W. Cheung, Y. Shin, D. M. Bortz, D. Ghosh, *et al.*, "A comprehensive review of latent space dynamics identification algorithms for intrusive and non-intrusive reduced-order-modeling," arXiv preprint arXiv:2403.10748 (2024).

[17] S. G. Rosofsky and E. Huerta, "Magnetohydrodynamics with physics informed neural operators," arXiv preprint arXiv:2302.08332 (2023).

[18] Z. Bai, X. Wei, W. Tang, L. Oliker, Z. Lin, and S. Williams, "Ftl: Transfer learning nonlinear plasma dynamic transitions in low dimensional embeddings via deep neural networks," arXiv preprint arXiv:2404.17466 (2024).

[19] J. Kumar, D. Zarzoso, V. Grandgirard, J. Ebert, and S. Kesselheim, "Physics informed neural networks applied to the description of wave-particle resonance in kinetic simulations of fusion plasmas," (2023), arXiv:2308.12312 [physics.comp-ph].

[20] J. Qiu, J. Huang, X. Zhang, Z. Lin, M. Pan, Z. Liu, and F. Miao, "Pi-fusion: Physics-informed diffusion model for learning fluid dynamics," (2024), arXiv:2406.03711 [physics.flu-dyn].

[21] G. Dong, X. Wei, J. Bao, G. Brochard, Z. Lin, and W. Tang, "Deep learning based surrogate models for first-principles global simulations of fusion plasmas," Nuclear Fusion **61**, 126061 (2021).

[22] J. Kates-Harbeck, A. Svyatkovskiy, and W. Tang, "Predicting disruptive instabilities in controlled fusion plasmas through deep learning," Nature **568**, 526–531 (2019).

[23] X. Wei, S. Sun, W. Tang, Z. Lin, H. Du, and G. Dong, "Reconstruction of tokamak plasma safety factor profile using deep learning," Nuclear Fusion **63**, 086020 (2023).

[24] V. Gopakumar, S. Pamela, L. Zanisi, Z. Li, A. Gray, D. Brennand, N. Bhatia, G. Stathopoulos, M. Kusner, M. P. Deisenroth, *et al.*, "Plasma surrogate modelling using fourier neural operators," Nuclear Fusion **64**, 056025 (2024).

[25] R. Rossi, M. Gelfusa, A. Murari, and on behalf of JET contributors, "On the potential of physics-informed neural networks to solve inverse problems in tokamaks," Nuclear Fusion **63**, 126059 (2023).

[26] D. Gottlieb and S. A. Orszag, "10. efficient implementation of spectral methods," in *Numerical Analysis of Spectral Methods* (SIAM, Philadelphia, 1977) Chap. 10, pp. 117–120, https://epubs.siam.org/doi/pdf/10.1137/1.9781611970425.ch10.

[27] D. Schnack, D. Baxter, and E. Caramana, "A pseudospectral algorithm for three-dimensional magnetohydrodynamic simulation," Journal of Computational Physics **55**, 485–514 (1984).

[28] J. Kossaifi, N. Kovachki, K. Azizzadenesheli, and A. Anandkumar, "Multi-grid tensorized fourier neural operator for high-resolution pdes," arXiv preprint arXiv:2310.00120 (2023).

[29] Z. Li, D. Z. Huang, B. Liu, and A. Anandkumar, "Fourier neural operator with learned deformations for pdes on general geometries," Journal of Machine Learning Research **24**, 1–26 (2023).

[30] A. Tran, A. Mathews, L. Xie, and C. S. Ong, "Factorized fourier neural operators," arXiv preprint arXiv:2111.13802 (2021).

[31] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, and A. Anandkumar, "Physics-informed neural operator for learning partial differential equations," arXiv preprint arXiv:2111.03794 (2021).

[32] S. Cao, F. Brarda, R. Li, and Y. Xi, "Spectral-refiner: Fine-tuning of accurate spatiotemporal neural operator for turbulent flows," arXiv preprint arXiv:2405.17211 (2024).

[33] C. Sovinec and J. King, "Analysis of a mixed semi-implicit/implicit algorithm for low-frequency two-fluid plasma modeling," Journal of Computational Physics **229**, 5803–5819 (2010).

[34] J. R. King, A. Y. Pankin, S. E. Kruger, and P. B. Snyder, "The impact of collisionality, flr, and parallel closure effects on instabilities in the tokamak pedestal: Numerical studies with the nimrod code," Physics of Plasmas **23** (2016), 10.1063/1.4954302.

[35]E. C. Howell, J. R. King, S. E. Kruger, J. D. Callen, R. J. La Haye, and R. S. Wilcox, "Growing neoclassical tearing modes seeded via transient-induced-multimode interactions," Physics of Plasmas **29** (2022), 10.1063/5.0076253.

[36]D. D. Schnack, J. Cheng, D. C. Barnes, and S. E. Parker, "Comparison of kinetic and extended magnetohydrodynamics computational models for the linear ion temperature gradient instability in slab geometry," Physics of Plasmas **20** (2013), 10.1063/1.4811468.

[37]J. B. O'Bryan, C. R. Sovinec, and T. M. Bird, "Simulation of current-filament dynamics and relaxation in the pegasus spherical tokamak," Physics of Plasmas **19** (2012), 10.1063/1.4746089.

[38]G. Dong, J. Bao, A. Bhattacharjee, A. Brizard, Z. Lin, and P. Porazik, "Gyrokinetic particle simulations of the effects of compressional magnetic perturbations on drift-alfvenic instabilities in tokamaks," Physics of Plasmas **24** (2017).

[39]J. R. King, S. E. Kruger, K. H. Burrell, X. Chen, A. M. Garofalo, R. J. Groebner, K. E. J. Olofsson, A. Y. Pankin, and P. B. Snyder, "Mhd modeling of a diii-d low-torque qh-mode discharge and comparison to observations," Physics of Plasmas **24** (2017), 10.1063/1.4977467.

[40]A. Pankin, J. King, S. Kruger, X. Chen, K. Burrell, A. Garofalo, R. J. Groebner, G. McKee, and Z. Yan, "Towards validated mhd modeling of edge harmonic oscillation in diii-d qh-mode discharges," Nuclear Fusion **60**, 092004 (2020).

[41]X. S. Li, P. Lin, Y. Liu, and P. Sao, "Newly released capabilities in the distributed-memory superlu sparse direct solver," ACM Trans. Math. Softw. **49** (2023), 10.1145/3577197.