Towards Enforcing Hard Physics Constraints in Operator Learning

Valentin Duruisseaux [*1] **Miguel Liu-Schiaffini** [*1] **Julius Berner** [1] **Anima Anandkumar** [1]

Abstract

Enforcing physics constraints in surrogate models for PDE evolution operators can improve the physics plausibility of their predictions and their convergence and generalization properties. Imposing these differential constraints *softly* as training loss terms can suffer from various challenges and does not guarantee faithfulness to the constraints at inference time, calling for stronger ways to impose the constraints. In this paper, we strongly enforce physics constraints in operator learning by projecting the output of any given operator surrogate model onto the space of functions satisfying a specified differential constraint, and perform that projection in a suitable transformed space. Compared to prior works, our method is efficient, compatible with any existing operator learning architecture (both during or after training), and ensures that the physics constraint holds at all points in the spatiotemporal domain. While it remains unclear how to perform the projection efficiently for nonlinear differential constraints, we describe how our approach works remarkably well for linear differential constraints by performing the projection very efficiently in Fourier space. As an example, we enforce the divergence-free condition of the incompressible Navier–Stokes equations, where our projection operator enforces the constraint without sacrificing faithfulness to the data, and does so at a small additional cost.

1. Introduction and related works

Partial differential equations (PDEs) are a critical tool for modeling physical phenomena and dynamical systems relevant for scientific and engineering applications (Strang, 2007). Unfortunately, existing numerical methods for solving PDEs become very computationally expensive when applied to large-scale systems such as the Earth's climate (Schneider et al., 2017; 2019). As an attempt to circumvent these computational limitations, machine learning methods have recently been proposed to learn solution operators of PDEs from data (Lu et al., 2021a; Gupta & Brandstetter, 2022; Kovachki et al., 2023).

Among these methods, neural operators (Li et al., 2020b; Kovachki et al., 2023; Azizzadenesheli et al., 2024) have shown great promise, primarily due to their ability to receive input functions at arbitrary discretizations and query output functions at arbitrary points, but also due to their universal operator approximation property (Kovachki et al., 2021). A variety of neural operators have been proposed (Li et al.; Li et al., 2023; Rahman et al., 2022; 2023; Kossaifi et al., 2023; Liu-Schiaffini et al., 2023) and successfully applied to a wide range of problems (Kurth et al., 2022; Wen et al., 2023; Gopakumar et al., 2023; Zhou et al., 2024). Other operator learning frameworks exist, such as DeepONet (Lu et al., 2021a) and its extensions which have been applied in various contexts (Clark di Leoni et al., 2021; Lin et al., 2021; 2023; Yin et al., 2021; Oommen et al., 2022; Lanthaler et al., 2022; Duruisseaux & Chakraborty, 2023).

However, purely data-driven approaches may underperform in situations with limited or low-resolution data (Li et al., 2021). In these cases, data-driven models may be supplemented or constrained using prior knowledge of physics constraints or conservation laws (Karniadakis et al., 2021; Liu & Chowdhury, 2023). In operator learning, these constraints are typically incorporated as additional loss terms (Li et al., 2021; Wang et al., 2021b; Goswami et al., 2022; White et al., 2023), similarly to how they are enforced in Physics-Informed Neural Networks (PINNs) (Raissi et al., 2017a;b; 2019). However, a variety of issues can arise with these *soft* constraints. The resulting composite loss can be more challenging to minimize, with more expensive training iterations due to the presence of derivatives in the physics loss, and may be more prone to numerical issues (such as worse conditioning, conflicting loss terms, inaccuracies in approximating derivatives, and existence of trivial or non-desired solution satisfying the constraints on the collocation points where the physics loss is computed) (Li et al., 2021; Leiteritz & Pflüger, 2021; Krishnapriyan et al., 2021; Wang et al., 2021a; 2022; Rohrhofer et al., 2022; White et al., 2023). In

[*]Equal contribution [1]Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, CA 91125, USA. Correspondence to: Valentin Duruisseaux <vduruiss@caltech.edu>, Miguel Liu-Schiaffini <mliuschi@caltech.edu>, Julius Berner <jberner@caltech.edu>.

Accepted at *41st International Conference on Machine Learning* (ICML) 2024 AI for Science workshop, Vienna, Austria.

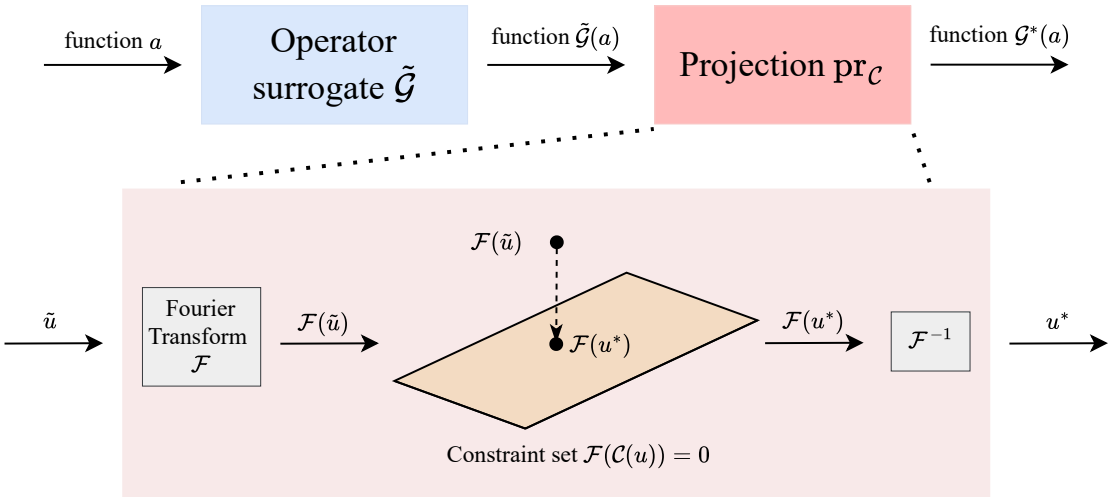


Figure 1. Diagram of our proposed framework for enforcing general physics constraints applied to linear differential physics constraints. Projecting onto the constraint set amounts to solving a least-squares problem with a linear constraint in Fourier space.

addition, the ability to provably and robustly follow known physical constraints is crucial for certain high-risk applications, but soft constraints are unable to provide guarantees of faithfulness to physics constraints at inference time.

These drawbacks motivate the exact enforcement of physics constraints in machine learning models, both at training and inference time, and a variety of approaches have been proposed for neural networks. Structural modifications to PINNs have been proposed to enforce hard Dirichlet and periodic boundary conditions (Lu et al., 2021b). ProbConserv (Hansen et al., 2024) enforces conservation laws by first obtaining a predictive distribution for the solution at specified target points and then applying a discretization of the integral form of the constraint as a Bayesian update. Harder et al. (2024) designed constraining layers to incorporate downscaling constraints into neural networks for climate super-resolution, to ensure mass and energy conservation between low and high resolution. Negiar et al. (2022) and Chalapathi et al. (2024) proposed to find optimal linear combinations of learned basis functions that satisfy the PDE at the given collocation points by solving a PDE-constrained optimization problem using differentiable optimization.

In the context of strongly imposing the divergence-free condition $\nabla \cdot u = 0$ for fluid flows (which we consider in our numerical experiments), Mohan et al. (2023) leveraged the fact that a vector field u is divergence-free if $u = \nabla \times v$ for some vector field v, and learned v using neural networks. Richter-Powell et al. (2022) explored the use of further characterizations of divergence-free vector fields to strongly enforce that constraint in neural networks. In a more general setting, HelmFluid (Xing et al., 2024) leverages the Helmholtz theorem (Bladel, 1959) that decomposes a dynamic field into a curl-free component and a divergence-free component $u = \nabla\Phi + \nabla \times v$ and learns the potential function Φ and the stream function v. However, these methods are in general harder to train or may lack expressivity because of the reparametrizations, and also rely on derivative computations which are typically costly. Jiang et al. (2020) introduced a spectral projection layer for neural networks and investigated its use with 3d CNNs. Jiang et al. (2020)'s approach serves as inspiration for the proposed approach, and can be thought of as the special case of our approach to enforce the divergence-free condition in neural networks.

More generally, hard constraints have also been enforced in structured dynamical systems for which the underlying structure is very well-understood, such as Hamiltonian systems with their underlying symplectic structure (Lutter et al., 2019; Zhong et al., 2020; Jin et al., 2020; Burby et al., 2020; Cranmer et al., 2020; Sæmundsson et al., 2020; Santos et al., 2022; Valperga et al., 2022; Duruisseaux et al., 2023a;b).

To the best of our knowledge, the only attempt at systematically enforcing hard constraints specifically for operator learning is BOON (Saad et al., 2023) which focuses on enforcing boundary conditions by modifying the operator kernels. However, BOON does not allow for enforcing physical constraints in the interior of the domain. As such, to our knowledge, our work is the first to enforce physical constraints within the interior of the physical domain in the operator learning setting.

Our approach: We enforce hard physics constraints in operator learning frameworks by projecting the output function of any operator surrogate model onto the space of functions satisfying a specified constraint, and perform this projection in a suitable transformed space.

Compared to prior works, our method is compatible with any

existing operator learning architecture (both during training and at inference time), and ensures that the physics constraint holds at all points in the spatiotemporal domain.

While it remains unclear how to perform the projection efficiently for nonlinear differential constraints, for linear differential constraints, the projection can be performed in Fourier space very efficiently, as presented by Jiang et al. (2020) with neural networks, with a small additional cost. By pairing the Fourier projection layer more generally with a neural operator, this additional layer can be thought of as a mapping on function spaces, and the output function can be queried on an arbitrarily fine grid. This way, with a neural operator, the discretization error of our approach can, in principle, be driven arbitrarily low. In addition, the operator learning paradigm allows carefully designed architectures, such as graph neural operators (GNOs) (Li et al., 2020a;c), to predict on arbitrary geometries with non-uniform grids, which could also be enabled by our approach after suitable modifications. Overall, thinking of the projection layer as a mapping on function spaces (instead of a mapping between grid discretizations) allows to leverage the numerous advantages of neural operators, and allows to consider new settings and can lead to further future developments and that were not initially conceivable with neural networks.

We test our approach for linear differential constraints to enforce the divergence-free condition for flow fields in incompressible Navier-Stokes, and investigate the effect of appending the projection layer during training or at inference time. In this context, the projection onto the space of divergence-free vector fields is also known as the Leray projection. Table 1 compares the desirable properties of our approach and other existing methods to enforce the divergence-free condition. We demonstrate experimentally that our projection layer enforces strongly the divergence-free condition at every point without harming the fidelity to the data. This complements the observations made by Jiang et al. (2020) that the projection layer can also lead to superior predictions of various key flow statistics. In addition, we demonstrate experimentally that the divergence error of our approach converges to 0 as the resolution of the discretization of the output function from the operator model increases. We also observe that it may not be necessary to perform the entire training procedure on the constrained model and that only finetuning the trained unconstrained model using the constraints for a few epochs might be better suited in certain cases. Unlike prior approaches, our projection layer can also be used in a plug-and-play fashion.

In our experiments, the benefits provided by the addition of our projection layer come at a computational time cost between $1.06\times$ and $1.16\times$ that of the baseline model.

Method	Efficient	Holds exactly	Res.-agnostic
Soft physics loss	✓	✗	✓
(Jiang et al., 2020)	✓	✓	✗
(Richter-Powell et al., 2022)	✗	✓	✓
(Mohan et al., 2023)	✓	✓	✗
Ours	✓	✓	✓

Table 1. Comparison of different methods for enforcing the divergence-free condition, the particular instance of a linear differential constraint we consider in our experiments.

2. Background and problem statement

Let $\mathcal{A}, \mathcal{U}$ be Banach spaces. Consider a parametric PDE

$$\begin{aligned} \mathcal{P}(a;u) &= 0 \quad && \text{in } \Omega \times (0,\infty) \\ \mathcal{B}(a;u) &= 0 \quad && \text{on } \delta\Omega \times (0,\infty) \\ \mathcal{I}(a;u) &= 0 \quad && \text{on } \bar{\Omega} \times \{0\}, \end{aligned} \tag{1}$$

where $\mathcal{P}, \mathcal{B}, \mathcal{I}$ are operators representing the PDE, boundary conditions, and initial conditions, respectively. The *operator learning problem* consists in learning the solution operator $\mathcal{G} : \mathcal{A} \to \mathcal{U}$ mapping an input function $a \in \mathcal{A}$ to the corresponding solution function $u \in \mathcal{U}$ of the PDE. The solution operator $\mathcal{G}$ of interest depends on the problem; one example is the time-stepping operator (Stuart & Humphries, 1998) for the solution $u(x,t)$, where $\mathcal{A} = \mathcal{U}$, which maps $u(x,0) \mapsto u(x,h)$ for some positive time-step $h > 0$.

We first consider general nonlinear differential constraints $\mathcal{C}(u) = 0$ on int $(\Omega) \times (0,\infty)$, and propose a general framework for projecting the output function of any operator surrogate model onto the space of functions satisfying these nonlinear differential constraints. While it is unclear how to construct an efficient projection for general nonlinear differential constraints, we will later restrict ourselves to the class of *linear differential constraints* of the form

$$0 = \mathcal{C}(u(x,t)) = \sum_{|\alpha| \le k} \ell_\alpha(x,t) \partial^\alpha u(x,t), \tag{2}$$

where α is an n-dimensional multi-index, for which the projection can be performed in Fourier space very efficiently.

Problem statement: Given the true solution operator $\mathcal{G}$, let $\tilde{\mathcal{G}}$ denote a surrogate model. In general, the output of $\tilde{\mathcal{G}}$ does not satisfy the specified physics differential constraints $\mathcal{C}(u) = 0$ over the interior of the domain. We would like to construct a mechanism to modify $\tilde{\mathcal{G}}$ and obtain a new surrogate model $\mathcal{G}^* : \mathcal{A} \to \mathcal{U}$ such that $\mathcal{G}^* \approx \tilde{\mathcal{G}}$ for which the constraints are enforced, i.e.,

$$\mathcal{C}\left(\mathcal{G}^*(a)\right) = 0 \quad \text{for all } a \in \mathcal{A}. \tag{3}$$

In practice, since a is a discretized function, we relax this condition and require that

$$\sup_{a \in \mathcal{A}} \mathcal{C}\left(\mathcal{G}^*(a)\right) < \varepsilon \tag{4}$$

for some tolerance $\varepsilon > 0$ depending only on the discretization of a, with $\varepsilon \to 0$ as this discretization is refined.

3. Proposed approach

3.1. General method

Given a surrogate model $\tilde{\mathcal{G}}$ for the solution operator $\mathcal{G}$, we define the constrained surrogate operator $\mathcal{G}^*$ via

$$\mathcal{G}^*(a) := (\mathrm{pr}_{\mathcal{C}} \circ \tilde{\mathcal{G}})(a), \tag{5}$$

where for $a \in \mathcal{A}$, $\mathrm{pr}_{\mathcal{C}}$ is an operator projecting $\tilde{\mathcal{G}}(a)$ onto the space of functions $u \in \mathcal{U}$ satisfying $\mathcal{C}(u) = 0$. The advantage of this decomposition is that the dependence on the constraint $\mathcal{C}$ lies only in $\mathrm{pr}_{\mathcal{C}}$. As a result, our approach is agnostic to the choice of surrogate model $\tilde{\mathcal{G}}$ and can be seamlessly combined with any operator learning framework.

The construction of $\mathrm{pr}_{\mathcal{C}}$ relies on a choice of invertible transform $\mathcal{T} : \mathcal{U} \to \mathcal{U}'$ with a corresponding discrete transform $\mathcal{T}_d$ (assuming without loss of generality that $\mathcal{T}(0) = 0$). With the transform $\mathcal{T}$, we can equivalently rewrite the linear differential constraint as $\mathcal{T}(\mathcal{C}(u)) = 0$.

Given $a \in \mathcal{A}$, a surrogate operator model $\tilde{\mathcal{G}}$, a discrete version $\tilde{u}_d$ of the model output $\tilde{u} = \tilde{\mathcal{G}}(a)$, and a transform $\mathcal{T}$ with its corresponding discrete transform $\mathcal{T}_d$, we construct the projection operator $\mathrm{pr}_{\mathcal{C}}$ as follows. We represent $\mathcal{T}(\tilde{u}_d)$ formally as

$$\mathcal{T}(\tilde{u}_d) = \sum_{k=1}^{K} c_k \varphi_k \tag{6}$$

in some specified K-dimensional function subspace with basis $\{\varphi_1, \ldots, \varphi_K\}$, where a good choice of K depends on the fineness of the discretization of $\tilde{u}_d$. We can approximate the constraint $\mathcal{T}(\mathcal{C}(\tilde{u}_d)) = 0$ as some algebraic equation

$$\hat{\mathcal{C}}(c_1, \ldots, c_K) = 0, \tag{7}$$

for the coefficients $\{c_1, \ldots, c_K\}$ of $\mathcal{T}(\tilde{u}_d)$.

The approximate projection of $\tilde{u}$ onto the space of functions of the form $\sum_{k=1}^{K} c_k \varphi_k$ satisfying $\hat{\mathcal{C}}(c_1, \ldots, c_K) = 0$ is obtained by solving the constrained optimization problem

$$\min_{c_1, \ldots, c_K} \left\| \tilde{u}_d - \mathcal{T}_d^{-1}\left(\sum_{k=1}^{K} c_k \varphi_k\right) \right\|_{\mathcal{U}'} \quad \text{subject to } \hat{\mathcal{C}}(c_1, , \ldots, c_K) = 0 \tag{8}$$

for the projected optimal coefficients $\{c_1^*, \ldots, c_K^*\}$, where the norm $\| \cdot \|_{\mathcal{U}'}$ is appropriately discretized given the discretization on $\tilde{u}_d$. The inverse discrete transform

$$u_d^* = \mathcal{T}_d^{-1}\left(\sum_{k=0}^{K} c_k^* \varphi_k\right) \tag{9}$$

then gives a discrete version u_d^* of the output of the constrained operator

$$u^* := \mathcal{G}^*(a) = (\mathrm{pr}_{\mathcal{C}} \circ \tilde{\mathcal{G}})(a). \tag{10}$$

Remark 3.1. In practice, many physics constraints of interest hold globally over the interior of the spatiotemporal domain. By applying the transform $\mathcal{T}$ and solving the optimization problem over $\{c_1, \ldots, c_K\}$, we ensure that the constraint holds approximately *over the entire domain*, with the discretization error decreasing as K increases.

To summarize, instead of enforcing the differential constraints directly in physical space where derivatives can be expensive to compute, we use a transform $\mathcal{T}$ and enforce the transformed constraints in the transformed space before returning to the original physical space. In practice, we choose the transform $\mathcal{T}$ so that the subsequent operations in the projection operator $\mathrm{pr}_{\mathcal{C}}$ become simpler or less computationally expensive. As an example, with linear differential constraints and $\mathcal{U} = \mathcal{U}' = L^2$, we can select the Fourier transform $\mathcal{F}$ and leverage Fourier theory to replace the linear differential constraints by a linear system of equations in Fourier modes, as discussed in Section 3.2.

Given a nonlinear differential constraint, we might not be able to find a transform $\mathcal{T}$ such that the constrained optimization problem (8) has a closed-form solution. In this case, we would need to rely on iterative optimization methods. For end-to-end training, one can leverage differentiable implicit layers (Amos & Kolter, 2017; Amos, 2019; Agrawal et al., 2019a;b) (also known as declarative nodes (Gould et al., 2021)) in which the relationship between the input and output is implicitly defined in terms of the solution to an optimization problem of a parametrized objective function. End-to-end learning is enabled via the implicit function theorem (instead of the chain rule) for efficient backpropagation without unrolling the optimization procedure. There are some similarities with the approach presented in Negiar et al. (2022); Chalapathi et al. (2024), where the constraints are enforced by adding a constraint layer made differentiable using implicit differentiation and differentiable optimization. However, in their approach they are not solving a constrained optimization problem in some transformed space, but rather solve a constrained optimization problem in the original space to find optimal linear combinations of learned basis functions that satisfy the PDE at given collocation points. They also need to fix the number of basis functions a priori and cannot use their layer in a plug-and-play fashion.

3.2. Application to linear differential constraints

We focus on the case where $\mathcal{C}$ is a linear differential constraint of the form

$$0 = \mathcal{C}(u(x,t)) = \sum_{|\alpha| \leq k} \ell_\alpha(x,t)\partial^\alpha u(x,t), \tag{11}$$

where, for $x \in \mathbb{R}^n$, each α is a n-dimensional multi-index. There are many applications governed by PDEs with linear differential constraints. Examples in fluid dynamics include the divergence-free condition (e.g., incompressible Navier-Stokes), the irrotational flow condition, and the continuity equation (e.g., compressible Navier-Stokes). A related constraint is the conservation of electrical charge in Maxwell's equations. Steady-state and equilibrium conditions in physical processes often take the form of linear differential constraints as well. Linear differential constraints include linear PDEs themselves (in this case, $\mathcal{C} = \mathscr{P}$), such as the Maxwell, Helmholtz, Klein–Gordon, Schrödinger, Poisson, Laplace, wave, heat, diffusion, and advection equations.

The proposed approach for enforcing linear differential constraints is depicted in Figure 1. We choose $\mathcal{T}$ to be the Fourier transform $\mathcal{F}$. Since differentiating in physical space corresponds to pointwise scalar multiplication in Fourier space, the discretized constraint in Equation (7) becomes a linear system of equations in the Fourier modes $\{c_1, \ldots, c_K\}$ of the discrete Fourier transform $\mathcal{F}_d$.

Furthermore, if $\mathcal{U} = \mathcal{U}' = L^2$, we can use the discrete form of Plancherel's theorem to rewrite the constrained optimization problem in Equation (8) as

$$\min_v \|\mathcal{F}_d(\tilde{u}_d) - v\|_{L^2} \quad \text{subject to } \hat{\mathcal{C}}v = 0, \tag{12}$$

where $v \in \mathbb{C}^k$ and $\hat{\mathcal{C}}$ is the constraint matrix. This is a least norm problem with linear constraint $\hat{\mathcal{C}}(v) = 0$, which has a closed-form solution via the pseudo-inverse $\hat{\mathcal{C}}^\dagger$ of $\hat{\mathcal{C}}$:

$$v^* = \mathcal{F}_d(\tilde{u}_d) - \hat{\mathcal{C}}^\dagger(\mathcal{F}_d(\tilde{u}_d)). \tag{13}$$

For $a \in \mathcal{A}$, we then define the projected output to be

$$\mathcal{G}^*(a) = \mathcal{F}_d^{-1}(v^*). \tag{14}$$

Remark 3.2. The closed form solution (13) of the least norm problem is differentiable, so the constrained operator $\mathcal{G}^*$ in Equation (14) is differentiable using auto-differentiation, given a differentiable surrogate model $\tilde{\mathcal{G}}$. The projection operator $\text{pr}_\mathcal{C}$ may thus be incorporated seamlessly into training pipelines and the constrained model can be trained end-to-end. In (Jiang et al., 2020), the constrained optimization problem is formulated slightly differently, and a more explicit formula for the projection in Fourier space is provided for the 3d case, used in conjunction with CNNs.

Remark 3.3. While Jiang et al. (2020) focused on neural networks (especially CNNs), the Fourier projection for linear differential constraints can be used more generally with a neural operator as surrogate model. In this context, the projection layer can be thought of as a mapping on function spaces, for which the output function is truly divergence-free. In practice, the divergence error of our method only depends on the resolution of the grid on which the output solution is queried, and can in principle be driven arbitrarily low by increasing the output resolution. In Section 4.2.1, we demonstrate empirically the convergence to 0 of the divergence with a constrained neural operator as the output resolution increases. In addition, the operator learning paradigm allows carefully designed architectures, such as graph neural operators (GNOs) (Li et al., 2020a;c), to predict on arbitrary geometries with non-uniform grids. With an adequate modification (e.g. a non-uniform FFT) and when paired with a GNO, our projection layer would then allow an end-to-end constrained operator surrogate model that can be queried at any location and for which the discretization error can be driven arbitrarily low.

4. Numerical Experiments

The projection method introduced in Section 3.2 can be applied to arbitrary linear differential constraints. In this paper, we focus on enforcing the divergence-free condition of flow fields in incompressible Navier–Stokes equations, with various Reynolds numbers. In this context, the projection onto the space of divergence-free vector fields is known as the Leray projection. We consider Kolmogorov flows, which can be expressed as

$$\partial_t u + u \cdot \nabla u - \frac{1}{\text{Re}}\Delta u = -\nabla p + \sin(4y)\hat{x} \tag{15}$$

$$\nabla \cdot u = 0, \tag{16}$$

where u is the velocity fields, p is the pressure field, Re denotes the Reynolds number, and $\hat{x}$ is a unit vector. These incompressible Navier–Stokes equations hold over a 2d toroidal domain $[0, 2\pi]^2$.

The solution operator of interest maps the initial condition $u(\cdot, 0) = u_0$ to a time-evolved solution $u(\cdot, \tau)$ for some time-step $\tau > 0$. The divergence-free constraint we enforce is $\nabla \cdot u = 0$, which is a linear differential operator. Details regarding the implementation of that constraint within our approach is presented in Appendix B. For our experiments, we use the $\text{Re} = 500$ and $\text{Re} = 5000$ Kolmogorov flows datasets presented in (Li et al., 2022), and refer the reader to that paper for more details regarding these datasets.

To investigate the effect of appending the projection layer during training or at inference time, we compare the performance of a baseline model with 3 different models:

- **Baseline**: the original surrogate model $\tilde{\mathcal{G}}$ is trained for N epochs in a purely data-driven way.
- **B+Projection**: (Baseline with Projection) The original surrogate model $\tilde{\mathcal{G}}$ is trained for N epochs in a purely data-driven way, and its output is projected using $\text{pr}_{\mathcal{C}}$ at inference time.
- **B+Finetuning**: (Baseline with Constraint Finetuning) After training the surrogate model $\tilde{\mathcal{G}}$ for $(N - N')$ epochs, we add the projection layer $\text{pr}_{\mathcal{C}}$ and finetune the constrained model $\mathcal{G}^* = (\text{pr}_{\mathcal{C}} \circ \tilde{\mathcal{G}})$ for N' epochs.
- **Fully Constrained**: We add the projection layer to the untrained surrogate model $\tilde{\mathcal{G}}$, and train the constrained model $\mathcal{G}^* = (\text{pr}_{\mathcal{C}} \circ \tilde{\mathcal{G}})$ for N epochs.

In this paper, we use a Fourier Neural Operator (FNO) (Li et al., 2020b) as our surrogate model $\tilde{\mathcal{G}}$. A general description of the FNO architecture is provided in Appendix A. The details regarding the experiments setup, the FNO architectures used, and the training hyperparameters, are presented in Appendix C.

Remark 4.1 (Computational Cost). For the divergence-free condition, the constraint matrix $\hat{\mathcal{C}}$ is the same for every problem given a fixed number of constraint modes. Thus, the the pseudo-inverse $\hat{\mathcal{C}}^\dagger$ of $\hat{\mathcal{C}}$ to solve the constrained least norm problem (12) only needs to be computed once. The overall additional cost of our approach (in the forward pass) lies in evaluating equations (13-14), which essentially amounts in our implementation to the cost of 2 matrix-vector multiplications, one Fast Fourier Transform (FFT), and one inverse FFT. On our computing architecture, with the size of our FNO and 64 constraint modes, each epoch to train the constrained FNO took between $1.06\times$ and $1.16\times$ the amount of time it took for one epoch with the unconstrained FNO (and the implementation of the projection could be further optimized).

Remark 4.2. We train each model with 5 different random seeds, and report our results as *mean (± standard deviation)*.

Remark 4.3. The projection layer can be thought of as a mapping on function spaces, for which the output function is divergence-free. However, some divergence error is introduced when discretizing the output function, and depends on the resolution of the discretization. This discretization error in our approach can, in principle, be driven arbitrarily low by querying the output function on a sufficiently fine grid. We demonstrate this in Section 4.2.1, by showing that the divergence error in the constrained model predictions decrease from $\sim 10^{-5}$ for a 64×64 resolution down to $\sim 10^{-11}$ for 1920×1920. For this reason, we report the divergence error in the constrained models predictions as ≈ 0 in the subsequent tables. Note that to compute derivatives to test the divergence error in the model predictions, we use spectral derivatives (i.e. we compute the derivatives in Fourier space) instead of finite differences derivatives since these allow us to get much more accurate derivatives at a lower computational cost.

Remark 4.4. The vector fields from the datasets considered here are not perfectly divergence-free, so the test L2 error can serve as a proxy for fidelity to the data but slightly higher values do not necessarily reflect poorer performance.

4.1. Re $= 500$ Kolmogorov Flow

We first consider the Re $= 500$ Kolmogorov flow dataset. The results, presented in Table 2, show that for the three different constrained approaches lead to a strongly enforced divergence-free condition without harming the fidelity to the data compared to the baseline FNO model. On the contrary, the projection layer consistently lead to lower test L2 error. Figure 2 displays an example of vector field predictions using the baseline model and the constrained approach.

Method	Test L2 ($\times 10^{-2}$)	Divergence
Baseline	2.3267 ($\pm$0.0924)	0.722 ($\pm$0.015)
B+Projection	2.2896 ($\pm$0.0917)	$\approx$ **0**
B+Finetuning	2.2278 ($\pm$0.1166)	$\approx$ **0**
Fully Constrained	**2.1574** ($\pm$0.0386)	$\approx$ **0**

Table 2. Test L2 and divergence-free errors obtained with the three methods based on our approach and the baseline model for the Re $= 500$ Kolmogorov flow dataset. See Remark 4.3 for an explanation of what is meant here by ≈ 0 divergence-free error.

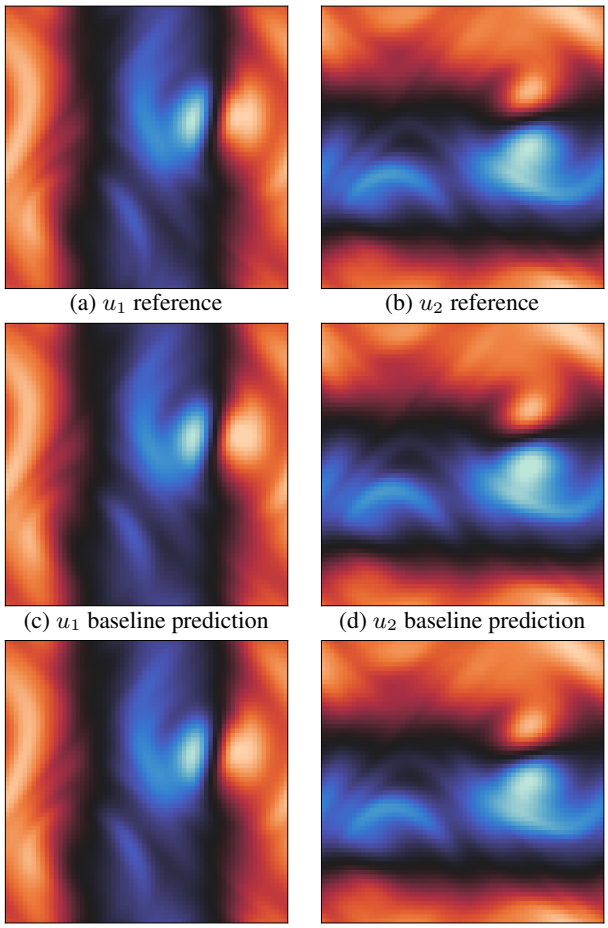
4.2. Re $= 5000$ Kolmogorov Flow

4.2.1. 64×64 Resolution

We now spatially subsample the 256×256 Re $= 5000$ Kolmogorov flow dataset to obtain 64×64 vector fields. The results, presented in Table 3, show that our approach leads to a strongly enforced divergence-free condition without hurting the fidelity to the data.

Method	Test L2 ($\times 10^{-2}$)	Divergence
Baseline	2.1535 ($\pm$0.0410)	1.641 ($\pm$0.015)
B+Projection	**2.0957** ($\pm$0.0390)	$\approx$ **0**
B+Finetuning	2.1006 ($\pm$0.0286)	$\approx$ **0**
Fully Constrained	2.1794 ($\pm$0.0238)	$\approx$ **0**

Table 3. Test L2 and divergence-free errors obtained with the three methods based on our approach and the FNO baseline model, for the Re $= 5000$ Kolmogorov flow dataset with resolution 64×64. See Remark 4.3 for an explanation of what is meant here by ≈ 0.



(a) u_1 reference (b) u_2 reference (c) u_1 baseline prediction (d) u_2 baseline prediction (e) u_1 constrained prediction (f) u_2 constrained prediction

Figure 2. Example of vector fields from the Re $= 500$ Kolmogorov flow test dataset with the corresponding predictions using the baseline model and the constrained approach.

We demonstrate that the divergence error obtained using the constraint projection can be driven arbitrarily low by increasing the resolution of the discretization of the output function. We evaluate the models on finer grids to obtain higher resolution vector fields and compute the divergence error using spectral derivatives on the finer-grid predictions. We start from a low resolution of 64×64 (i.e. scaling factor of 1), and progressively increase the resolution via $(64 \times S) \times (64 \times S)$ (i.e. scaling factor of S) up to 1920×1920 with a scaling factor of 30. We can see from Figure 3 the convergence to 0 of the divergence error for the constrained model predictions as the grid resolution increases, from $\sim 10^{-5}$ for 64×64 down to $\sim 10^{-11}$ for 1920×1920.

4.2.2. 128×128 Resolution

We repeat the experiment of Section 4.2.1 but subsample the Re $= 5000$ Kolmogorov flow dataset to obtain 128×128 vector fields. The results are presented in Table 4. In all 3 different methods, our approach led to a strongly enforced divergence-free condition with slightly lower test L2 errors.

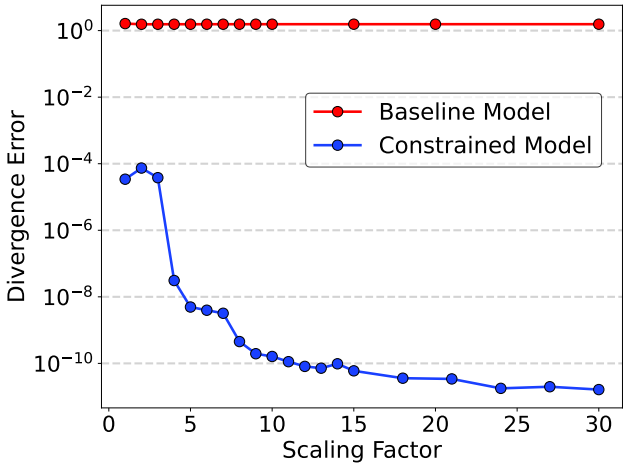


Figure 3. Evolution of the divergence error for the baseline and constrained models predictions as the grid resolution of the output, $(64 \times S) \times (64 \times S)$, increases with the scaling factor S.

Method	Test L2 ($\times 10^{-2}$)	Divergence
Baseline	1.6938 ($\pm$0.0470)	1.642 ($\pm$0.036)
B+Projection	1.6617 ($\pm$0.0440)	$\approx$ **0**
B+Finetuning	1.6722 ($\pm$0.0278)	$\approx$ **0**
Fully Constrained	**1.6367** ($\pm$0.0120)	$\approx$ **0**

Table 4. Test L2 and divergence-free errors obtained with the three methods based on our approach and the baseline model for the Re $= 5000$ Kolmogorov flow dataset with resolution 128×128. See Remark 4.3 for an explanation of what is meant here by ≈ 0.

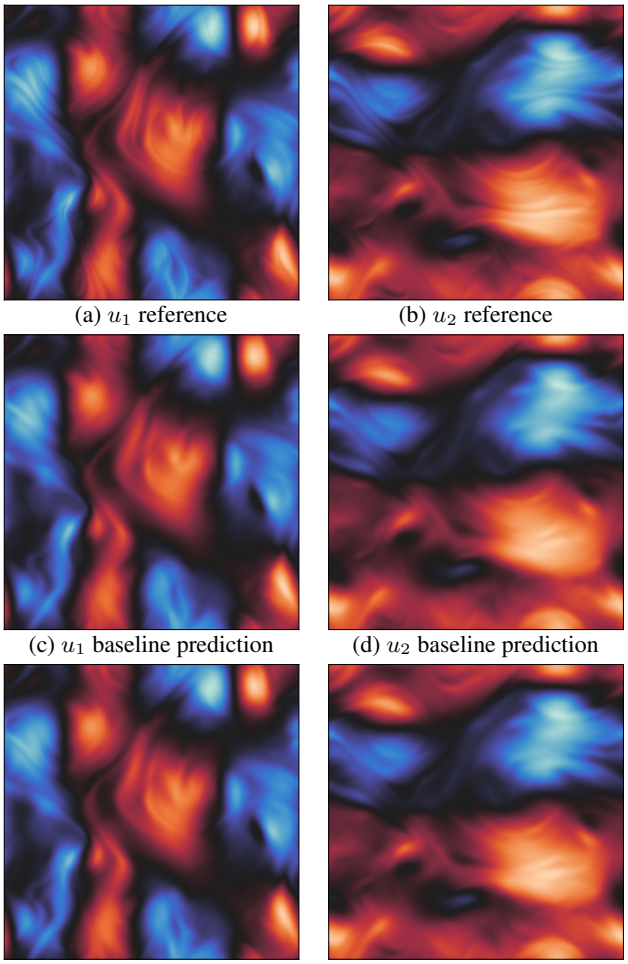
4.2.3. Zero-Shot Super-Resolution

Finally, we tested the ability of our projected surrogate models to perform zero-shot super-resolution. We used the models trained on 64×64 data from Section 4.2.1, and took advantage of the operator learning paradigm to predict the vector fields on a finer 256×256 grid.

Table 5 displays the L2 error when comparing the zero-shot super-resolution predictions to the reference Kolmogorov flow data, and Figure 4 displays an example of super-resolution predictions. The observations made in the earlier experiments extend to this super-resolution experiment: using the projection layer does not harm the test L2 error.

Method	$\times 4$ Super-Resolution Error ($\times 10^{-2}$)
Baseline	2.1351 ($\pm$0.0410)
B+Projection	**2.0804** ($\pm$0.0387)
B+Finetuning	2.0821 ($\pm$0.0285)
Fully Constrained	2.1580 ($\pm$0.0234)

Table 5. Test L2 super-resolution errors on 256×256 vector fields from the Re $= 5000$ Kolmogorov flow test dataset, with the baseline and constrained models trained on 64×64 data.



(a) u_1 reference (b) u_2 reference

(c) u_1 baseline prediction (d) u_2 baseline prediction

(e) u_1 constrained prediction (f) u_2 constrained prediction

Figure 4. Example of super-resolution results on 256×256 vector fields from the Re $= 5000$ Kolmogorov flow test dataset with the corresponding predictions using the baseline and constrained models trained on 64×64 data.

Discussion

In this paper, we have introduced a new general approach for enforcing hard physics constraints in operator learning frameworks, consisting in projecting the output of any operator surrogate model onto the space of functions satisfying a specified constraint, where the projection is performed in a suitable transformed space. We focus mostly on the special case of linear differential constraints, where the projection can be performed in Fourier space at a small cost.

We have seen throughout our experiments with incompressible Navier–Stokes equations that our projection layer strongly enforced the divergence-free condition on the predicted vector fields without harming the fidelity to the data compared to the baseline. It actually most often lead to lower test L2 errors, despite the divergence-free condition not being strongly enforced in the training data. This complements the numerical experiment conducted by Jiang et al. (2020) using neural networks which showed that the projection layer can also allow for superior results in terms of the distribution predictions of key flow statistics (such as total kinetic energy, dissipation, and large eddy turnover time).

In some cases, training the projected operator model with the projection layer led to further reduction of the L2 test errors when compared to projecting the trained baseline model (B+Projection). We also note that the variability of the results obtained, characterized by the standard deviation, seems to decrease as the number of epoch with the constraining layer increases. Improvement in the quality of the distribution predictions of various key flow statistics was also observed in the experiment conducted by Jiang et al. (2020) when the projection layer was included within a CNN during training. It may however not be necessary to carry the entire training on the constrained model (Fully Constrained). Only finetuning the trained unconstrained model using the constraints for a few epochs (B+Finetuning) might be more suitable in some contexts. For instance, in the low-data regime, training with the constraining projection right from the start could be harder than only using the projection layer for finetuning the trained unconstrained model once its predictions already match the data reasonably well.

We also demonstrated experimentally that the divergence error of the constrained models using our Fourier projection layer converges to 0 as the resolution of the discretization of the output function from the operator model increases. We further noted that the benefits provided by our projection layer come at minor additional computational cost.

While our approach performed very well for imposing the divergence-free condition on the fluid flow problems considered here, it suffers from limitations that need to be addressed before it can be used for more realistic larger-scale problems with limited available data. We intend to use this framework as a stepping stone towards a more general mechanism for enforcing hard constraints in operator learning frameworks. In particular, we will investigate how to extend our approach to allow for the full flexibility of using non-uniform output grids with graph neural operators and for non-periodic vector fields, and to integrate it with other mechanisms to enforce boundary conditions. Thinking of the projection layer as a mapping on function spaces (instead of a mapping between grid discretizations) allows to leverage the numerous advantages of neural operators over neural networks and could lead to numerous future developments and that were not initially conceivable with neural networks. Note that the application of the proposed approach to general nonlinear differential constraints remains unclear and unexplored, so we intend to pursue the construction of appropriate transforms and projection maps for specific restricted classes of nonlinear differential constraints.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., and Kolter, J. Z. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems*, volume 32, 2019a.

Agrawal, A., Barratt, S. T., Boyd, S. P., Busseti, E., and Moursi, W. M. Differentiating through a cone program. *Journal of Applied and Numerical Optimization*, 2019b.

Amos, B. *Differentiable Optimization-Based Modeling for Machine Learning*. PhD thesis, Carnegie Mellon University, May 2019.

Amos, B. and Kolter, J. Z. OptNet: Differentiable optimization as a layer in neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 136–145. PMLR, 06–11 Aug 2017.

Azizzadenesheli, K., Kovachki, N., Li, Z., Liu-Schiaffini, M., Kossaifi, J., and Anandkumar, A. Neural operators for accelerating scientific simulations and design. *Nature Reviews Physics*, pp. 1–9, 2024.

Bladel, J. On Helmholtz's theorem in finite regions. *IRE Transactions on Antennas and Propagation*, 7(5):119–119, 1959. doi: 10.1109/TAP.1959.1144767.

Burby, J. W., Tang, Q., and Maulik, R. Fast neural Poincaré maps for toroidal magnetic fields. *Plasma Physics and Controlled Fusion*, 63(2):024001, dec 2020.

Chalapathi, N., Du, Y., and Krishnapriyan, A. S. Scaling physics-informed hard constraints with mixture-of-experts. In *The Twelfth International Conference on Learning Representations*, 2024.

Clark di Leoni, P., Lu, L., Meneveau, C., Karniadakis, G., and Zaki, T. DeepONet prediction of linear instability waves in high-speed boundary layers. 2021.

Cranmer, M., Greydanus, S., Hoyer, S., Battaglia, P. W., Spergel, D. N., and Ho, S. Lagrangian neural networks. *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Duruisseaux, V. and Chakraborty, A. An operator learning framework for spatiotemporal super-resolution of scientific simulations. 2023.

Duruisseaux, V., Burby, J. W., and Tang, Q. Approximation of nearly-periodic symplectic maps via structure-preserving neural networks. *Scientific Reports*, 13(8351), 2023a.

Duruisseaux, V., Duong, T., Leok, M., and Atanasov, N. Lie group forced variational integrator networks for learning and control of robot systems. In *Proceedings of the 5th Annual Learning for Dynamics and Control Conference*, volume 211 of *Proceedings of Machine Learning Research*, pp. 731–744. PMLR, 2023b.

Gopakumar, V., Pamela, S., Zanisi, L., Li, Z., Anandkumar, A., and Team, M. Fourier neural operator for plasma modelling. *arXiv preprint arXiv:2302.06542*, 2023.

Goswami, S., Bora, A., Yu, Y., and Karniadakis, G. E. Physics-informed deep neural operator networks. 2022. doi: 10.48550/arXiv.2207.05748.

Gould, S., Hartley, R., and Campbell, D. J. Deep declarative networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. ISSN 1939-3539. doi: 10.1109/tpami.2021.3059462.

Gupta, J. K. and Brandstetter, J. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

Hansen, D., Maddix, D. C., Alizadeh, S., Gupta, G., and Mahoney, M. W. Learning physical models that can respect conservation laws. *Physica D: Nonlinear Phenomena*, 457:133952, 2024. ISSN 0167-2789. doi: 10.1016/j.physd.2023.133952.

Harder, P., Hernandez-Garcia, A., Ramesh, V., Yang, Q., Sattigeri, P., Szwarcman, D., Watson, C., and Rolnick, D. Hard-constrained deep learning for climate downscaling, 2024.

Jiang, C. M., Kashinath, K., Prabhat, and Marcus, P. Enforcing physical constraints in CNNs through differentiable PDE layer. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Jin, P., Zhang, Z., Zhu, A., Tang, Y., and Karniadakis, G. E. SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. *Neural Networks*, 132(C), 12 2020.

Karniadakis, G. E., Kevrekidis, I. G., Lu, L., Perdikaris, P., Wang, S., and Yang, L. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

Kossaifi, J., Kovachki, N., Azizzadenesheli, K., and Anandkumar, A. Multi-grid tensorized fourier neural operator for high resolution PDEs. 2023.

Kovachki, N., Lanthaler, S., and Mishra, S. On universal approximation and error bounds for Fourier neural operators. *J. Mach. Learn. Res.*, 22(1), 2021. ISSN 1532-4435.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Krishnapriyan, A. S., Gholami, A., Zhe, S., Kirby, R. M., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. In *Neural Information Processing Systems*, 2021.

Kurth, T., Subramanian, S., Harrington, P., Pathak, J., Mardani, M., Hall, D., Miele, A., Kashinath, K., and Anandkumar, A. FourCastNet: Accelerating global high-resolution weather forecasting using adaptive Fourier neural operators. 2022. doi: 10.48550/arXiv.2208.05419.

Lanthaler, S., Molinaro, R., Hadorn, P., and Mishra, S. Nonlinear reconstruction for operator learning of PDEs with discontinuities. 2022. doi: 10.48550/arXiv.2210.01074.

Leiteritz, R. and Pflüger, D. How to avoid trivial solutions in physics-informed neural networks. *ArXiv*, abs/2112.05620, 2021.

Li, Z., Huang, D. Z., Liu, B., and Anandkumar, A. Fourier neural operator with learned deformations for pdes on general geometries. *J. Mach. Learn. Res.*, 24(1), mar . ISSN 1532-4435.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations, 2020a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020b.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Multipole graph neural operator for parametric partial differential equations, 2020c.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *ACM/JMS Journal of Data Science*, 2021.

Li, Z., Liu-Schiaffini, M., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Learning chaotic dynamics in dissipative systems. *Advances in Neural Information Processing Systems*, 35: 16768–16781, 2022.

Li, Z., Kovachki, N., Choy, C., Li, B., Kossaifi, J., Otta, S. P., Nabian, M. A., Stadler, M., Hundt, C., Azizzadenesheli, K., and Anandkumar, A. Geometry-Informed Neural Operator for Large-Scale 3D PDEs. 2023.

Lin, C., Li, Z., Lu, L., Cai, S., Maxey, M., and Karniadakis, G. E. Operator learning for predicting multiscale bubble growth dynamics. *The Journal of Chemical Physics*, 154 (10), 03 2021. ISSN 0021-9606. doi: 10.1063/5.0041203.

Lin, G., Moya, C., and Zhang, Z. B-DeepONet: An enhanced bayesian DeepONet for solving noisy parametric PDEs using accelerated replica exchange SGLD. *Journal of Computational Physics*, 473:111713, 2023. ISSN 0021-9991. doi: 10.1016/j.jcp.2022.111713.

Liu, F. and Chowdhury, A. Deep learning with physics priors as generalized regularizers. *arXiv preprint arXiv:2312.08678*, 2023.

Liu-Schiaffini, M., Singer, C. E., Kovachki, N., Schneider, T., Azizzadenesheli, K., and Anandkumar, A. Tipping point forecasting in non-stationary dynamics on function spaces. *arXiv preprint arXiv:2308.08794*, 2023.

Lu, L., Jin, P., Pang, G., Zhang, Z., and Karniadakis, G. Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021a.

Lu, L., Pestourie, R., Yao, W., Wang, Z., Verdugo, F., and Johnson, S. G. Physics-informed neural networks with hard constraints for inverse design. *SIAM Journal on Scientific Computing*, 43(6):B1105–B1132, 2021b. doi: 10.1137/21M1397908.

Lutter, M., Ritter, C., and Peters, J. Deep Lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, 2019.

Mohan, A. T., Lubbers, N., Chertkov, M., and Livescu, D. Embedding hard physical constraints in neural network coarse-graining of three-dimensional turbulence. *Phys. Rev. Fluids*, 8:014604, Jan 2023. doi: 10.1103/PhysRevFluids.8.014604.

Negiar, G., Mahoney, M. W., and Krishnapriyan, A. S. Learning differentiable solvers for systems with hard constraints. 2022.

Oommen, V., Shukla, K., Goswami, S., Dingreville, R., and Karniadakis, G. E. Learning two-phase microstructure evolution using neural operators and autoencoder architectures. *npj Computational Materials*, 8, 09 2022. doi: 10.1038/s41524-022-00876-7.

Rahman, M. A., Florez, M. A., Anandkumar, A., Ross, Z. E., and Azizzadenesheli, K. Generative adversarial neural operators. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

Rahman, M. A., Ross, Z. E., and Azizzadenesheli, K. U-NO: U-shaped neural operators. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *ArXiv*, abs/1711.10561, 2017a.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *ArXiv*, abs/1711.10566, 2017b.

Raissi, M., Perdikaris, P., and Karniadakis, G. 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. ISSN 0021-9991. doi: 10.1016/j.jcp.2018.10.045.

Richter-Powell, J., Lipman, Y., and Chen, R. T. Q. Neural conservation laws: A divergence-free perspective. In *Advances in Neural Information Processing Systems*, 2022.

Rohrhofer, F., Posch, S., Gößnitzer, C., and Geiger, B. Understanding the difficulty of training physics-informed neural networks on dynamical systems. 2022.

Saad, N., Gupta, G., Alizadeh, S., and Maddix, D. C. Guiding continuous operator learning through physics-based boundary constraints. In *International Conference on Learning Representations*, 2023.

Sæmundsson, S., Terenin, A., Hofmann, K., and Deisenroth, M. P. Variational integrator networks for physically structured embeddings. In *AISTATS*, 2020.

Santos, S., Ekal, M., and Ventura, R. Symplectic momentum neural networks - using discrete variational mechanics as a prior in deep learning. In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, volume 168 of *Proceedings of Machine Learning Research*, pp. 584–595, Jun 2022.

Schneider, T., Teixeira, J., Bretherton, C. S., Brient, F., Pressel, K. G., Schär, C., and Siebesma, A. P. Climate goals and computing the future of clouds. *Nature Climate Change*, 7(1):3–5, 2017.

Schneider, T., Kaul, C. M., and Pressel, K. G. Possible climate transitions from breakup of stratocumulus decks under greenhouse warming. *Nature Geoscience*, 12(3): 163–167, 2019.

Strang, G. *Computational science and engineering*. SIAM, 2007.

Stuart, A. and Humphries, A. R. *Dynamical systems and numerical analysis*, volume 2. Cambridge University Press, 1998.

Valperga, R., Webster, K., Turaev, D., Klein, V., and Lamb, J. Learning reversible symplectic dynamics. In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, volume 168 of *Proceedings of Machine Learning Research*, pp. 906–916. PMLR, Jun 2022.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021a. doi: 10.1137/20M1318043.

Wang, S., Wang, H., and Perdikaris, P. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science Advances*, 7(40): eabi8605, 2021b. doi: 10.1126/sciadv.abi8605.

Wang, S., Yu, X., and Perdikaris, P. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022. ISSN 0021-9991. doi: 10.1016/j.jcp.2021.110768.

Wen, G., Li, Z., Long, Q., Azizzadenesheli, K., Anandkumar, A., and Benson, S. M. Real-time high-resolution CO2 geological storage prediction using nested Fourier neural operators. *Energy Environ. Sci.*, 16:1732–1741, 2023. doi: 10.1039/D2EE04204E.

White, C., Berner, J., Kossaifi, J., Elleithy, M., Pitt, D., Leibovici, D., Li, Z., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operators with exact differentiation on arbitrary geometries. In *The Symbiosis of Deep Learning and Differential Equations III*, 2023.

Xing, L., Wu, H., Ma, Y., Wang, J., and Long, M. Helmfluid: Learning helmholtz dynamics for interpretable fluid prediction. In *International Conference on Machine Learning*, 2024.

Yin, M., Ban, E., Rego, B., Zhang, E., Cavinato, C., Humphrey, J., and Karniadakis, G. E. Simulating progressive intramural damage leading to aortic dissection using an operator-regression neural network. 2021.

Zhong, Y. D., Dey, B., and Chakraborty, A. Symplectic ODE-Net: Learning Hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2020.

Zhou, T., Wan, X., Huang, D. Z., Li, Z., Peng, Z., Anandkumar, A., Brady, J. F., Sternberg, P. W., and Daraio, C. Ai-aided geometric design of anti-infection catheters. *Science Advances*, 10(1):eadj1741, 2024.

A. Description of FNO

Neural operators compose linear integral operators $\mathcal{K}$ with pointwise non-linear activation functions σ to approximate highly non-linear operators. More precisely, we define the **neural operator**

$$\mathcal{G}_\theta := \mathcal{Q} \circ (W_L + \mathcal{K}_L + b_L) \circ \cdots \circ \sigma(W_1 + \mathcal{K}_1 + b_1) \circ \mathcal{P} \tag{17}$$

where $\mathcal{P}$, $\mathcal{Q}$ are the pointwise neural networks that encode the lower dimension function into higher dimensional space and vice versa. The model stacks L layers of $\sigma(W_l + \mathcal{K}_l + b_l)$ where W_l are pointwise linear operators (matrices), $\mathcal{K}_l$ are integral kernel operators, b_l are bias terms, and σ are fixed activation functions. The parameters θ consists of all the parameters in $\mathcal{P}, \mathcal{Q}, W_l, \mathcal{K}_l, b_l$.

A **Fourier neural operator (FNO)** (Li et al., 2020b) is a neural operator using *Fourier integral operator* layers, which are defined via

$$\big(\mathcal{K}(\phi)v_t\big)(x) = \mathcal{F}^{-1}\Big(R_\phi \cdot (\mathcal{F}v_t)\Big)(x) \tag{18}$$

where R_ϕ is the Fourier transform of a periodic function κ parameterized by ϕ. On a uniform mesh, the Fourier transform $\mathcal{F}$ can be implemented using the fast Fourier transform (FFT). Here is a depiction of the Fourier Neural Operator:

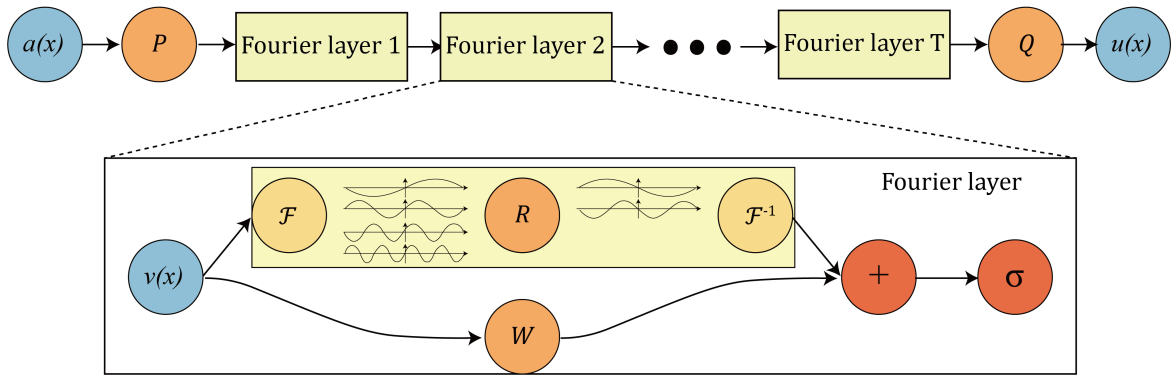


Figure 5. The FNO architecture (extracted from (Li et al., 2020b)).

B. Divergence-free Constraint Implementation

Given a velocity field $u(x) = (u_1(x), u_2(x))$ on the 2d torus $x = (x_1, x_2) \in \mathbb{T}^2$, the divergence-free condition reads

$$\nabla \cdot u(x_1, x_2) = \frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2} = 0. \tag{19}$$

Although we consider a time-evolving flow field in the Navier–Stokes equation, we remove the dependence on time here for notational convenience since the divergence-free condition applies at all times.

Since u is assumed to be a 2d periodic function, its non-zero Fourier modes $\xi = (\xi_1, \xi_2)$ lie in $\mathbb{Z}^2$. Taking the Fourier transform of Equation (19) gives

$$\mathcal{F}\left(\frac{\partial u_1}{\partial x_1} + \frac{\partial u_2}{\partial x_2}\right)(\xi_1, \xi_2) = 0 \qquad \Longrightarrow \qquad \xi_1 \hat{u}_1(\xi) + \xi_2 \hat{u}_2(\xi) = 0, \tag{20}$$

where $\hat{u}$ denotes the Fourier transform of u. This constraint must hold for all $(\xi_1, \xi_2) \in \mathbb{Z}^2$.

Upon discretizing into a $K \times K$ grid and applying the discrete Fourier transform instead of the continuous Fourier transform, Equation (20) remains the same, with the exception that ξ now takes values $\{c_0, \ldots, c_{K-1}\}^2$, for some discrete set of modes $c_0, \ldots, c_{K-1}$. We may thus construct the constraint matrix $\hat{C}$ from Equation (7) by adding one row representing Equation (20) for every such ξ. This ensures that the divergence-free condition applies over the entire domain, up to discretization error.

However, there is an additional practical concern: the current construction does not guarantee that the solution to Equation (8) is a purely real velocity field, and naively removing the imaginary components after projection may result in a flow field with non-zero divergence. As such, we must enforce conjugate symmetry. Specifically, for every non-Nyquist (ξ_1, ξ_2), the following must hold:

$$\hat{u}_1(\xi_1, \xi_2) = \hat{u}_1(-\xi_1, -\xi_2) \qquad \hat{u}_2(\xi_1, \xi_2) = \hat{u}_2(-\xi_1, -\xi_2). \tag{21}$$

Furthermore, at the Nyquist frequencies, $\hat{u}_1$ and $\hat{u}_2$ must be purely real.

To enforce both the conjugate symmetry and real Nyquist criterion, we decompose each Fourier mode into its real and imaginary components which makes both the variables and the constraints in the optimization problem purely real.

C. Hyperparameters

In our numerical experiments, we used a Fourier Neural Operator (FNO) with 6 layers, each of width 64 with 20 Fourier modes. The resulting FNO model possesses 10.9M trainable parameters. For the constrained models, we used 64 Fourier constraint modes for the projection layer.

We trained all our models for $N = 800$ epochs and used $N' = 50$ for the B+Finetuning approach. The models were trained using the following optimizer and schedulers in PyTorch:

```
optimizer = torch.optim.Adam(model.parameters(), lr=0.002, weight_decay=1e-6)

scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer, 'min', factor = 0.8, patience= 20)
```

The models were trained on 90% of the data, and tested on the remaining 10%.

For the Re $= 500$ Kolmogorov flow experiment from Section 4.1, the dataset consists of 1000 simulations of spatial resolution 64×64 with 400 time steps each, for a total of 400,000 samples. We used a batch size of 256 in that experiment.

For the Re $= 5000$ Kolmogorov flow experiments from Section 4.2, the dataset consists of 100 simulations of spatial resolution 256×256 with 400 time steps each, for a total of 40,000 samples. The dataset is downsampled in space to obtain the 64×64 and 128×128 datasets. We used 256 as batch size for the 64×64 data, and 32 for the 128×128 data.