Learning to correct spectral methods for simulating turbulent flows

Anonymous authors
Paper under double-blind review

Abstract

Despite their ubiquity throughout science and engineering, only a handful of partial differential equations (PDEs) have analytical, or closed-form solutions. This motivates a vast amount of classical work on numerical simulation of PDEs and more recently, a whirlwind of research into data-driven techniques leveraging machine learning (ML). A recent line of work indicates that a hybrid of classical numerical techniques with machine learning can offer significant improvements over either approach alone. In this work, we show that the choice of the numerical scheme is crucial when incorporating physics-based priors. We build upon Fourier-based spectral methods, which are considerably more efficient than other numerical schemes for simulating PDEs with smooth and periodic solutions. Specifically, we develop ML-augmented spectral solvers for three model PDEs of fluid dynamics, which improve upon the accuracy of standard spectral solvers at the same resolution. We also demonstrate a handful of key design principles for combining machine learning and numerical methods for solving PDEs.

1 Introduction

The numerical simulation of nonlinear partial differential equations (PDEs) permeates science and engineering, from the prediction of weather and climate to the design of engineering systems. Unfortunately, solving PDEs on the fine grids required for high-fidelity simulations is often unfeasible due to its prohibitive computational cost. This leads to inevitable trade-offs between runtime and accuracy. The status quo is to solve PDEs on grids that are coarse enough to be computationally feasible but often too coarse to resolve all phenomena of interest. One classical approach is to derive coarse-grained surrogate PDEs such as Reynold's Averaged Navier Stokes (RANS) and Large Eddy Simulation (LES) (Pope, 2000), which in principle can be accurately solved on coarse grids. This family of approaches has enjoyed widespread success but is tedious to perform, PDE-specific, and suffers from inherent accuracy limitations (Durbin, 2018; Pope, 2004).

Machine learning (ML) has the potential to overcome many of these limitations by inferring coarse-grained models from high-resolution ground-truth simulation data. Turbulent fluid flow is an application domain that has already reaped some of these benefits. Pure ML methods have achieved impressive results, in terms of accuracy, on a diverse set of fluid flow problems (Li et al., 2021; Stachenfeld et al., 2022). Hybrid methods, which combine classical numerical simulation with ML, have achieved improved stability and generalization in addition to accuracy (Kochkov et al., 2021; List et al., 2022). However, the progress made by hybrid methods has been limited to low order finite difference and finite volume methods, with the exception of one recent study (Frezat et al., 2022).

Beyond finite differences and finite volumes, there is a broad field of established numerical methods for solving PDEs. In this paper, we focus on spectral methods, which are used throughout computational physics (Trefethen, 2000; Burns et al., 2020) and constitute the core of the current state-of-the-art weather forecasting system (Roberts et al., 2018). Spectral methods, when applicable, are often preferred over other numerical methods because they can be more accurate for equations with smooth solutions. In fact, their accuracy rivals that of the recent progress made by ML. This begs the question: *Can we improve spectral solvers of turbulent fluid flows using learned corrections of coarse-grained simulations?*

Our contributions are as follows:

1. We propose a hybrid physics machine learning method that provides sub-grid corrections to classical spectral methods.
2. We explore two toy 1D problems, the Unstable Burgers' equation and the Kuramoto-Sivashinsky equation, demonstrating precisely the cases where the hybrid model is able to make the largest improvements.
3. We compare spectral, finite-volume, ML-only, and our hybrid model on a 2D forced turbulence task. Our hybrid models improve upon the accuracy of spectral-only methods, which already show remarkable enhanced performance compared to recent work. Furthermore, our results show that a key modeling choice for both hybrid and pure ML models is the use of velocity- rather than vorticity-based state representations.

1.1 Related work

The study of turbulent fluid dynamics is vast. We refer to Pope (2000) for a thorough introduction. Classical approaches tend to derive mathematical approximations to the governing equations in an *a priori* manner. Recently, there has been an explosion of work in data-driven methods at the interface of computational fluid dynamics (CFD) and machine learning (ML). We loosely organize this recent work into three main categories:

Purely Learned Surrogates fully replace numerical schemes from CFD with purely learned surrogate models. A number of different architectures have been explored, including multi-scale convolutional neural networks (Ronneberger et al., 2015; Wang et al., 2020), graph neural networks (Sanchez-Gonzalez et al., 2020), and Encoder-Process-Decoder architectures (Stachenfeld et al., 2022).

Operator Learning seeks to learn the differential operator directly by mimicking the analytical properties of its class, such as pseudo-differential or Fourier integral operators but without explicit physics-informed components. These methods often leverage the Fourier transform (Li et al., 2021; Tran et al., 2021) and the off-diagonal low-rank structure of the associated Green's function (Fan et al., 2019; Li et al., 2020).

Hybrid Physics-ML is a recent line of work which seeks to hybridize classical numerical methods with contemporary data-driven deep learning techniques (Mishra, 2019; Bar-Sinai et al., 2019; Kochkov et al., 2021; List et al., 2022; Frezat et al., 2022; Bruno et al., 2021) These approaches seek to *learn* corrections to basic numerical schemes from high-resolution simulation data. The goal is to take the best of both worlds — the simplicity of PDE-based governing equations, with the expressive power of neural networks. Working in this vein, our goal is learn corrections to spectral methods which we introduce below. In contrast to a recent prior study of ML for augmenting spectral solvers for closure models relevant to climate modeling (Frezat et al., 2022), we focus on key ML architecture choices and include comparisons to the baseline of using classical numerical methods on different resolution grids.

2 Spectral methods for fluids

Spectral methods are a powerful method for finding high-accuracy solutions to PDEs. They are often the method of choice for solving smooth PDEs with simple boundary conditions and geometries. There is an extensive literature on the theoretical and practical underpinnings of spectral methods (Gottlieb & Orszag, 1977; Gottlieb & Hesthaven, 2001; Canuto et al., 2007; Kopriva, 2009), particularly for methods based on Fourier spectral collocation (Trefethen, 2000; Boyd, 2001), which we succinctly introduce here.

2.1 Partial differential equations for turbulent fluids

Let $\mathbf{u} : \mathbb{R}^d \times \mathbb{R}^+ \to \mathbb{R}^{d'}$ be a time-varying vector field, for dimensions d and d'. We study PDEs of the form

$$\partial_t \mathbf{u} = D\mathbf{u} + N(\mathbf{u}), \tag{1}$$

plus initial and boundary conditions. Here D is a linear partial differential operator, and N is a non-linear term. Equations of this form dictate the temporal evolution of $\mathbf{u}$ driven by its variation in space. In practice,

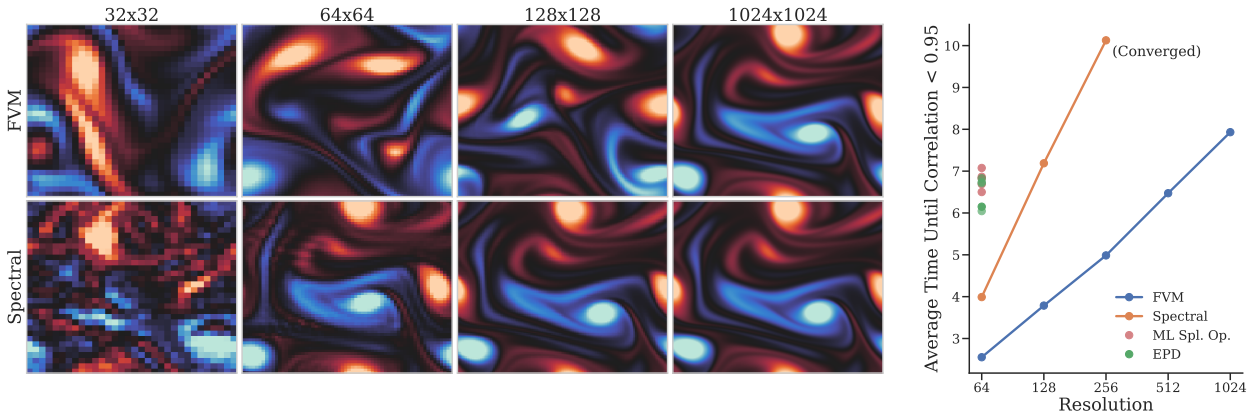

Figure 1: Comparing finite volume (FVM) to spectral convergence for 2D turbulence. *Left*: Vorticity fields evolved up to time 9.0 with different grid sizes and numerical methods, starting from an identical initial condition. Qualitatively, it is clear that at resolution 64x64, the spectral method has already captured most of features of the high-resolution state. The FVM looks sharp, but clearly differs. At sufficiently high resolution, the methods converge to the same solution. *Right*: Here we compare each method with high-resolution 2048x2048 ground truth. We plot the time until the first dip below 0.95 of the correlation with the ground truth. Note that the initial conditions for the finite volume and the spectral method are sampled from the same distribution, but are not identical. At resolution 256, the spectral method has converged within the accuracy limitations of single precision, so we omit higher resolutions. For EPD baseline and our ML Split Operator (ML Spl. Op.) models, we show performance across five different neural network parameter initializations.

PDEs are solved by discretizing in space and time, which converts the continuous PDE into a set of update rules for vectors of coefficients to approximate the state $\mathbf{u}$ in some discrete basis, e.g., on a grid. For time-dependent PDEs, temporal resolution must be scaled proportionally to spatial resolution to retain an accurate solution, so runtime for PDE solvers scales at best like $O(n^{d+1})$, where d is the number of spatial dimensions and n is the number of discretization points along any dimension.

For simulations of fluids, the differential operator is typically either diffusive, $D = \partial_x^2$, or hyper-diffusive, $D = \partial_x^4$; and the non-linearity is a convective term, $N = \frac{1}{2}\partial_x(\mathbf{u}^2) = \mathbf{u}\partial_x\mathbf{u}$. *Diffusion* is the tendency of the fluid velocity to become uniform due to internal friction, and *convection* is the tendency of the fluid to be transported by its own inertia. Turbulent flows are characterized by a convective term that is much larger than diffusion.

For most turbulent flows of interest, closely approximating the exact PDE (known as Direct Numerical Simulation) is computationally intractable, due to the high required grid resolution. Instead, coarse-grained approximations of the PDE are solved, known as "Large Eddy Simulation" (LES). LES augments Equation (1) by adding a correction term given by a closure model to account for averaging over fine spatial length scales. In practice, this term is often omitted due to the difficulty of deriving appropriate formulas ("implicit LES") and the PDE is solved with whatever grid resolution is computationally feasible. In our case, correction terms are an opportunity for machine learning. If we can accurately model PDEs on coarser grids with suitable correction terms, we may be able to significantly reduce the computational cost of large-scale simulations.

2.2 The appeal of the Fourier basis for modeling PDEs

Let us further assume that $\mathbf{u}(x,t) : [0, 2\pi] \times \mathbb{R}^+ \to \mathbb{R}$ in Equation (1) is one-dimensional, 2π-periodic and square-integrable for all times t. Consider the Fourier coefficients $\widehat{\mathbf{u}}^t$ of $\mathbf{u}(x,t)$, truncated to lowest $K+1$ frequencies (for even K):

$$\widehat{\mathbf{u}}^t = (\widehat{\mathbf{u}}^t_{-K/2}, \ldots, \widehat{\mathbf{u}}^t_k, \ldots, \widehat{\mathbf{u}}^t_{K/2}) \quad \text{where} \quad \widehat{\mathbf{u}}^t_k = \frac{1}{2\pi}\int_0^{2\pi} \mathbf{u}(x,t)e^{-ik\cdot x}dx. \tag{2}$$

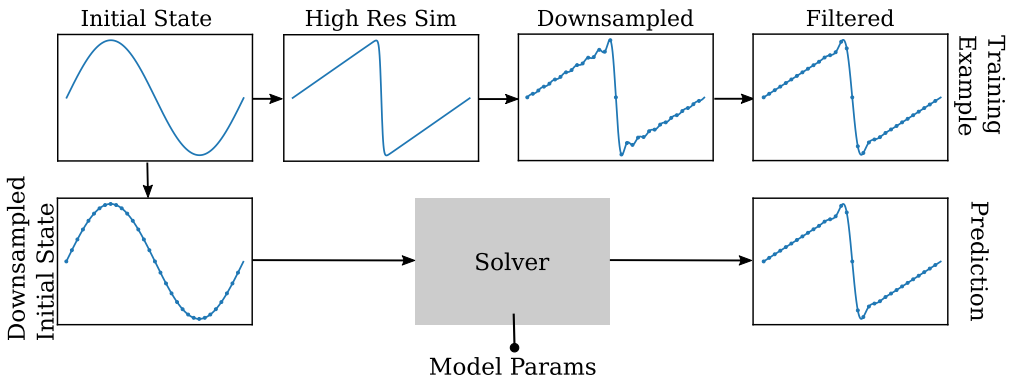

Figure 2: Diagram of our training pipeline. Starting with a high resolution initial state, we run it forward using a high-resolution spectral solver. Then we downsample by truncating higher frequencies in the Fourier representation. This can cause "ringing effects" for which the standard approach is to apply a filtering operator. Finally, we train a solver to mimic this process as closely as possible, as measured in ℓ_2-loss.

$\widehat{\mathbf{u}}^t$ is a vector representing the solution at time t, containing the coefficients of the Fourier basis e^{ikx} for $k \in \{-K/2, \ldots, K/2\}$. In general, the integral $\widehat{\mathbf{u}}_k^t$ has no analytical solution and so we approximate it using a trapezoidal quadrature on $K+1$ points. Namely, we approximate it by sampling $\mathbf{u}(x,t)$ on an equispaced grid. The Fast Fourier Transform (FFT) (Cooley & Tukey, 1965) computes these Fourier coefficients efficiently in log-linear time. Spectral methods for PDEs leverage the fact that differentiation in the Fourier domain can be calculated by element-wise multiplication according to the identity $\partial_x \widehat{\mathbf{u}}_k = ik\widehat{\mathbf{u}}_k$. This makes inverting linear differential operators easy since it is simply element-wise division (Trefethen, 2000).

When time-dependent PDEs include non-linear terms, spectral methods evaluate these terms on a grid in real-space, which requires forward and inverse FFTs inside each time-step. These transforms introduce two sources of error. First, the quadrature rules for $\widehat{\mathbf{u}}_k^t$ produce only an approximation to the integral. Second, there will be a truncation error when the number of frequencies $K+1$ is less than the bandwidth of $\mathbf{u}(x,t)$. Remarkably, it is a well-known fact of Fourier analysis that both of these errors decay super-algebraically if $\mathbf{u}(x,t)$ is periodic and is in C^∞ (Trefethen, 2000).[1] Thus, relatively few discretization points are needed to represent solutions which are C^∞.

Because of these favorable convergence properties, spectral methods often outshine their finite difference counterparts. For example, spectral methods are used for large-scale simulations of turbulence on GPU super-computers (Yeung & Ravikumar, 2020). See Figure 1 for a simple comparison in the case of 2D turbulence. This motivates us to start from spectral methods, rather than finite difference methods, and further improve them using machine learning.

3 Learned split operators for correcting spectral methods

Similarly to classical numerical methods, we solve Equation (1) by representing our state in a finite Fourier basis as $\widehat{\mathbf{u}}^t$ and integrating forward in time. We model the fully known differential operators D and N using a standard (pseudo-)spectral method, which we denote here as Physics-Step. Our machine learning approach, described here, is built upon insight used by traditional physics-based solvers.

We use a simple explicit-Euler integrator in time for the correction yielding the following recurrence relation:

$$\widehat{\mathbf{u}}^{t+1} = \text{Physics-Step}(\widehat{\mathbf{u}}^t) + h \cdot \text{Learned-Correction}(\widehat{\mathbf{u}}^t; \theta), \tag{3}$$

where $h \in \mathbb{R}$ is the time step size. Learned-Correction$(\cdot; \theta)$ contains tune-able parameters θ, which are optimized during training to match high-resolution simulations.

[1] Convergence is exponential for analytic $\mathbf{u}$ (Tadmor, 1986).

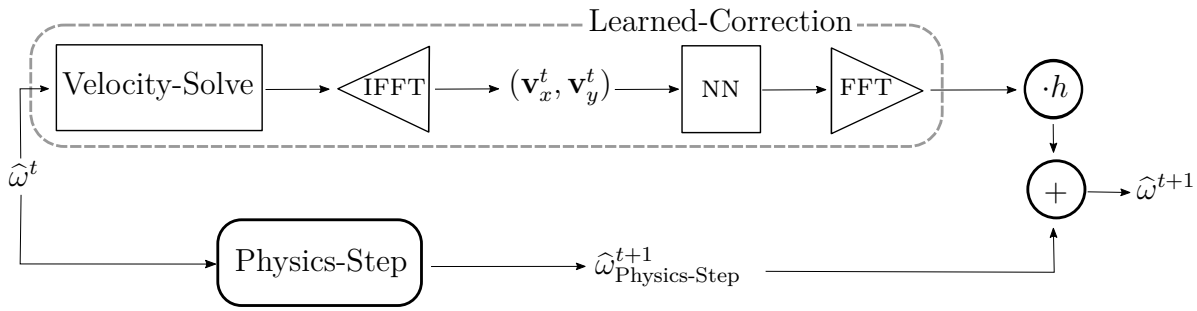

Figure 3: Diagram summarizing our model described in Equation (3) for the 2D Navier-Stokes equation. The input, vorticity $\widehat{\omega}^t$, is processed by two independent components — Learned-Correction and Physics-Step — operating at different time-scales. The output of Learned-Correction is weighted by h, as in a basic first-order Euler stepping scheme, and combined with the output of the Physics-Step to give the state at the next time step.

3.1 Convolutional layers for encoding the physical prior of locality

We leverage the physical intuition that information travels at finite speed — for a small time step h, the solution at time $t + h$ only depends locally in space on the solution at time t (within a dependency cone usually encoded by the Courant–Friedrichs–Lewy (CFL) condition). Following previous work (see Sec. 1.1, Hybrid Physics-ML), we incorporate this assumption into the machine learning component of our model using Convolutional Neural Networks (CNN or ConvNet) which, by design, only learn local features.

Since the Fourier basis is global, each coefficient $\widehat{\mathbf{u}}_k^t$ contains information from the full spatial domain as shown in Equation (2). Thus, in order to maintain spatial features, which are local, we apply the ConvNet component of our Neural Split Operator model in real-space. This is accomplished efficiently via IFFT and FFT to map the signal back and forth between frequency- and real-space.

Since we always work with periodic boundary conditions, all of our ConvNets use periodic padding. For both 1D and 2D problems we used an Encoder-Process-Decoder (EPD) architecture for the NN component (Stachenfeld et al., 2022). See Section 4.1 for a description of EPD models and Appendix C for further detail.

3.2 The split operator method for combining time scales

Different numerical methods can be more suitable for time-advancing different terms in PDEs, due to a variety of considerations including relative accuracy requirements, numerical stability and computational feasibility. Accordingly, *split operator methods* (Strang, 1968; McLachlan & Quispel, 2002) that combine different time integrators are a popular tool for solving PDEs, which we also find to be useful when evaluating extra terms given by neural networks.

In the usual pattern of spectral methods, Physics-Step itself is split into two components corresponding to the D and N terms of Equation (1). The D term of Physics-Step is solved with an implicit Crank-Nicolson method and the N part is solved using explicit 4th-order Runge-Kutta, which effectively runs at a time-step of $h/4$. The Learned-Correction component is solved using a vanilla first-order Euler time-step with step size h, which as we show in Section 4 is more accurate than putting the learned correction inside the 4th-order Runge-Kutta solver for 2D turbulence, in addition to being 4x less computationally expensive. Alternatively, omitting Physics-Step from Equation (3) gives a Neural ODE model (Chen et al., 2018) with first-order time-stepping, which we use for the pure-ML EPD models that serves as strong baselines in Section 4.

3.3 Physics-based solvers for calculating neural-net inputs

A final important consideration is the choice of input representation for the machine learning model. The most suitable representation for a given numerical solver may not necessarily be the most suitable representation for ML. Accordingly, we allow for using physics-based solvers to compute input variables used by the neural networks in our learned correction models. Our overall model is Learned-Correction $=$ $\textsc{fft}(\textsc{nn}(\textsc{ifft}(\text{State-Transform}(\widehat{\mathbf{u}}^t))))$, where $\textsc{nn}$ is implemented a periodic ConvNet. For our 1D test problems, State-Transform is the identity transformation, but for 2D Navier-Stokes (depicted in Fig. 3) we use State-Transform to calculate velocity from vorticity. This turns out to be key modeling choice, as described in Section 4.

3.4 Training and evaluation

Data preparation for spectral solvers. For ground truth data, we use fully resolved simulations, which we then downsample to the coarse target resolution. Choosing the downsampling procedure is a key decision for coarse-grained solvers (Frezat et al., 2022). In this work, the fully resolved trajectories are first truncated to the target wavenumber (i.e., an ideal low-pass filter). Then, we apply an exponential filter of the form $\tilde{\mathbf{u}}_k = \exp(-\alpha|k/k_{\max}|^{2p})\widehat{\mathbf{u}}_k$, where $\tilde{\mathbf{u}}_k$ denotes the filtered field and k is the k-th wavenumber (Canuto et al., 2006). We obtained stable trajectories using a relatively weak filter with $\alpha = 6$ and $p = 16$. The exponential filter smooths discontinuities in the PDE solution, which otherwise manifest themselves globally in real space as ringing artifacts known as Gibbs Phenomena (Canuto et al., 2006).

Filtering is also used to correct aliasing errors in spectral methods that arise when evaluating non-linear terms (Gottlieb & Hesthaven, 2001). Ideally, filtering for aliasing errors would be spatially adaptive (Boyd, 1996). In practice, however, filters for both aliasing and truncation are often chosen heuristically. While one might aspire to learn these heuristics from data, our attempts at doing so were unsuccessful, in part because insufficiently filtered simulations often entirely diverge rather than accumulating small errors. Thus, in addition to filtering the downsampled training data, we used the same exponential filter on the outputs of Physics-Step of Equation (3). Omitting this also resulted in global errors which were impossible for the learned component to correct. Figure 5 contains a failure mode on a model without filtering and Figure 2 summarizes our data generation pipeline, including an explicit filtering step.

Training loss. We train our models to minimize ℓ_2-loss over some number of unrolled prediction steps, which allows our model to account for compounding errors (Um et al., 2020). If $\overline{\mathbf{u}}(x,t)$ denotes our prediction at time t, then our training loss is given by $\beta \sum_{x,t\le T} |\overline{\mathbf{u}}(x,t) - \mathbf{u}(x,t)|^2$, with the scaling constant so that the loss of predicting "no change" is one, $\beta^{-1} = \sum_{x,t\le T} |\mathbf{u}(x,0) - \mathbf{u}(x,t)|^2$. In practice, we unroll over a relatively small number of time steps (e.g., $T = 32$ for 2D turbulence) during training, relative to the thousands of time-steps over which we hope simulations will be accurate, due to the decreased efficiency and stability of training over long time windows (Kochkov et al., 2021).

Measuring convergence. We seek to optimize the accuracy of coarse-grained simulations. Thus, we aim to make a coarse resolution simulation as similar as possible to a high-resolution simulation which has been coarse-grained in post-processing. Following Kochkov et al. (2021), we measure convergence to a fully resolved, high-resolution ground-truth at each time t in terms of mean absolute error (MAE), correlation, and time until correlation is less than 0.95. MAE is defined as $\sum_x |\overline{\mathbf{u}}(x,t) - \mathbf{u}(x,t)|$ and correlation is defined $\mathrm{Corr}[\mathbf{u}(\cdot,t), \overline{\mathbf{u}}(\cdot,t)] = \sum_x (\overline{\mathbf{u}}(x,t)\cdot\mathbf{u}(x,t))/(\|\overline{\mathbf{u}}(x,t)\|_2\|\mathbf{u}(x,t)\|_2)$ (since our flows have mean zero). Finally, we compute the first time the correlation dips below 0.95, i.e. $\min\{t \mid \mathrm{Corr}[\mathbf{u}(\cdot,t), \overline{\mathbf{u}}(\cdot,t)] < 0.95\}$. MAE is sensitive up to floating point precision but is not readily interpretable, whereas correlation is less sensitive but more interpretable.

4 Results

We showcase our method using three model equations which capture many of the algorithmic difficulties present in more complex systems: one-dimensional Kuramoto–Sivashinsky (KS) equation, one-dimensional

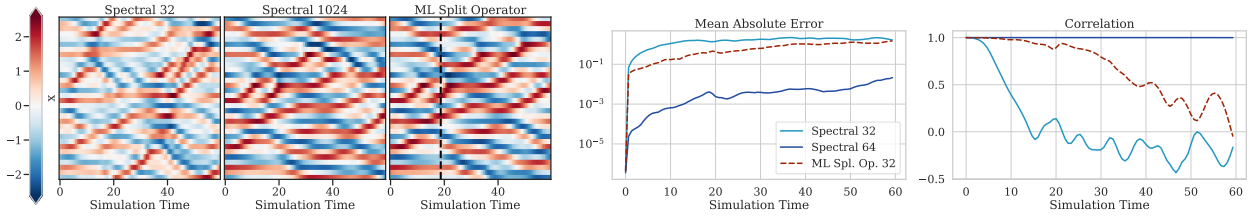

Figure 4: Comparing our model to spectral-only baselines on the KS equation. The spectral method is essentially converged to a 1024 ground truth model at a resolution of 64. Vertical dashed line indicates the first time step in which our model's correlation with the ground truth is less than 0.9. Our model is still able to some improvement over a coarse resolution (Spectral 32) baseline.

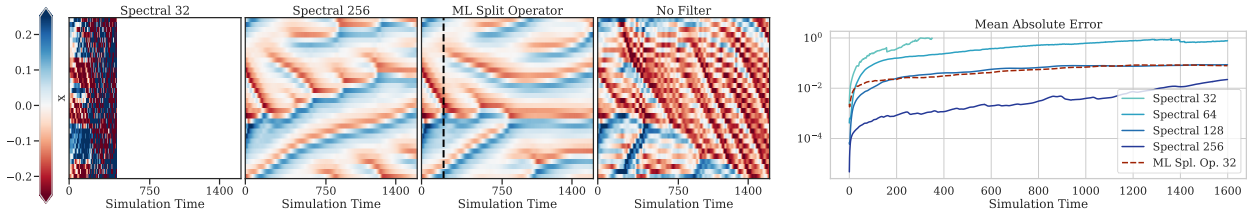

Figure 5: Comparing our model to spectral-only baselines on the unstable Burgers' equation (Eq. (5)). Vertical dashed line indicates the first time step in which our model's correlation with the ground truth is less than 0.9. Not only is our model stable unlike the coarse (Spectral 32) baseline, it also performs on par with a medium-grain (128) baseline, demonstrating a 4x improvement in spatial accuracy. If the Physics-Step component of Equation (3) does not include the exponential filter as described in Section 3.4, the Learned-Correction component diverges completely (*Left* panel, "No Filter").

unstable Burgers' equation, and two-dimensional Kolmogorov flow, a variant of incompressible Navier-Stokes with constant forcing. The KS equation has smooth solutions which spectral methods are well-designed to solve. Unstable Burgers' equation presents a test case in which classical spectral methods struggle. Finally, with two-dimensional Kolmogorov flow, we aim to make a demonstration on a more realistic problem.

4.1 Baselines

Our hybrid spectral-ML method naturally gives rise to two types of baselines: spectral-only and ML-only. On all three equations, we compare to spectral-only at various resolutions, e.g., "Spectral 32" refers to the spectral-only method with 32 grid points.

For Kolmogorov flow, we compare to two additional baselines: (1) the Encoder-Process-Decoder (EPD) model (Stachenfeld et al., 2022), a state-of-the-art ML-only model, and (2) a finite volume method (FVM) as implemented by Kochkov et al. (2021) — a standard numerical technique which serves as an alternative to the spectral method.

The EPD architecture acts in three steps. First, the spatial state is embedded into a high-dimensional space using a feed-forward neural network encoder architecture. Then, either another feed-forward or a recurrent architecture is applied to process the embedded state. Finally, the output of the process step is projected to back to the spatial state using another feed-forward decoder module. This method has recently achieved state-of-the-art on a wide range of one-, two-, and three-dimensional problems. For consistency, we used an identical EPD model for the NN component in our model (Eq. (3)).

Finally, we included a second-order FVM on a staggered grid to illustrate the strength of the spectral baselines. This gives context to compare to previous work in hybrid methods (Bar-Sinai et al., 2019; Kochkov et al., 2021) which use FVM as a baseline.

4.2 Kuramoto-Sivashinsky (KS) equation

The Kuramoto-Sivashinsky (KS) equation is a model of instability in flame fronts. In the PDE literature, it is popular as a model system that exhibits chaotic dynamics in a 1D PDE. The KS equation is defined as

$$\partial_t \mathbf{u} = -\mathbf{u}\partial_x \mathbf{u} - \partial_x^4 \mathbf{u} - \partial_x^2 \mathbf{u}. \tag{4}$$

The three terms on the right hand side of this equation correspond to convection, hyper-diffusion and anti-diffusion, which drives the system away from equilibrium.

Because solutions of the KS equations are smooth, spectral methods are able to capture the dynamics of the KS equation extremely well. At a resolution of 64, the simulation is already effectively converged to the limits of single precision arithmetic, i.e., it enjoys a perfect correlation with a high-resolution ground truth.

Considering how well-suited spectral methods are for modeling this equation, it is remarkable that our ML-Physics hybrid model is able to achieve any improvement at all. Looking qualitatively at the left panel of Figure 4, there is a clear improvement over the spectral 32 baseline.

4.3 Unstable Burgers' equation

Burgers' equation is a simple one-dimensional non-linear PDE which is used as a toy model of compressible fluid dynamics. The characteristic behavior of Burgers' equation is that it develops shock waves. Practitioners use this equation to test the accuracy of discretization schemes near discontinuities.

Like turbulent flows, the behavior of Burgers' equation is dominated by the convection, but unlike turbulent flows, it is not chaotic. Prior studies of ML applied to Burgers' equation imposed random forcings (Bar-Sinai et al., 2019; Um et al., 2020), but due to the non-choatic nature of the equation, discretization errors decay rather than compounding over time. Instead, we use an unstable viscous Burgers' equation which slightly amplifies low frequencies in order to make the dynamics chaotic. Sakaguchi (1999) provides the following definition:

$$\partial_t \mathbf{u} = -u\partial_x \mathbf{u} + \nu\partial_x^2 \mathbf{u} + \int_0^L g(x - x')\partial_x^2 \mathbf{u}(x')dx' \tag{5}$$

for viscosity $\nu > 0$ and domain size L. The three terms in this unstable Burgers' equation correspond to convection, diffusion and a scale-selective amplification of low-frequency signals. The convolution $g(x - x')$ amplifies low frequencies, decaying smoothly to zero as the frequency increases but is best described in the Fourier space, which we defer to Appendix A.

Shock waves are notoriously difficult to model using the Fourier basis due to the discontinuous solution (Sec. 3.4). Due to the mixing of length scales introduced by convection, these errors can quickly propagate to affect the overall dynamics. This can result in instability at low resolutions, evidenced in our results by an unstable coarse model (resolution 32). The inadequacy of spectral methods for solving this problem also explains why our method is able to achieve approximately 4x improvement in spatial resolution over a spectral baseline. Results are summarized in Figure 5.

4.4 2D turbulence

Finally, we consider 2D turbulence described by the incompressible Navier-Stokes equation with Kolmogorov forcing (Kochkov et al., 2021). This equation can be written either in terms of a velocity field or a scalar vorticity field $\omega := \partial_x \mathbf{v}_y - \partial_y \mathbf{v}_x$ (Boffetta & Ecke, 2012). Here we use a vorticity formulation, which is most convenient for spectral methods and avoids the need to separately enforce the incompressibility condition $\nabla \cdot \mathbf{v} = 0$. The equation is given by

$$\partial_t \omega = -\mathbf{v} \cdot \nabla\omega + \nu\nabla^2\omega - \alpha\,\omega + f, \tag{6}$$

where the terms correspond to convection, diffusion, linear damping and a constant forcing f (App. B). The velocity vector field $\mathbf{u}((x, y), t)$ can be obtained from vorticity by solving a Poisson equation $-\nabla^2\psi = \omega$ for the stream function ψ and using the relation $\mathbf{v} = (\partial_y\psi, -\partial_x\psi)$. This velocity solve is computed via element-wise division and multiplication in the Fourier basis.

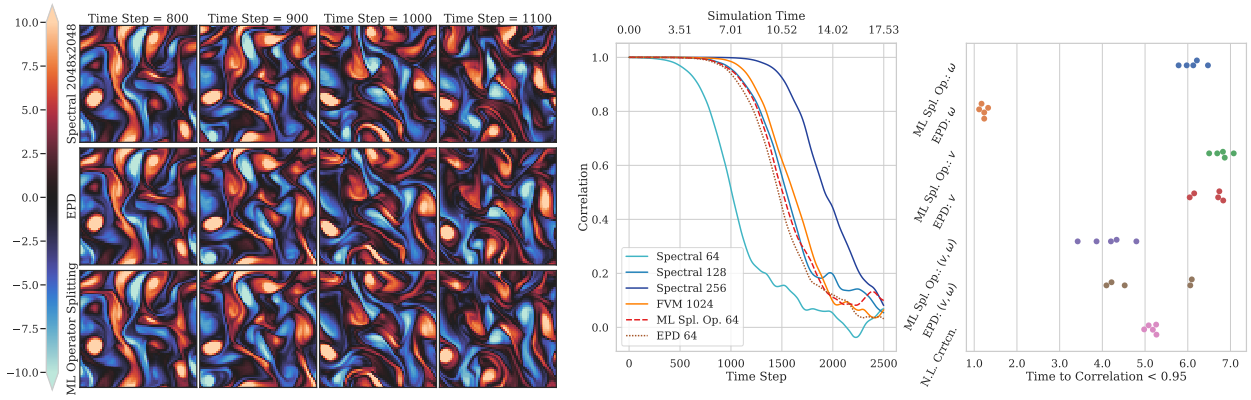

Figure 6: Benchmarking our Neural Split Operator (ML Spl. Op.) to spectral-only baselines and state-of-the-art ML-only, EPD model (Stachenfeld et al., 2022). *Left*: Visualizing model states starting at the time when divergence begins to occur. At time step = 800, EPD, Neural Split Operator model, and the high-resolution spectral-only 1024x1024 correlate well. By time step = 900, both learned models have begun to decorrelate with the ground truth albeit in different ways. *Right:* Comparing model variants across different random initializations. On the y-axis we measure the time to the first time step with correlation<0.95 with the high-resolution spectral ground truth. Then we compared our model with the EPD model across three data representation variations: velocity only ($\mathbf{v}$), vorticity only (ω) and velocity-vorticity concatenation ($(\mathbf{v}, \omega)$) as well as with an Nonlinear Term Correction model (N.L. Crrtn.). Shown here are the best performing models over five different random neural network parameter initializations.

For this problem, we compared four types of models: spectral-only, FVM-only, EPD (ML-only), and two types of hybrid methods: our split operator method and a nonlinear term correction method. The ML models are all trained to minimize the error of predicted velocities, which are obtained via a velocity solve for the spectral-only, EPD and hybrid models that represent state with vorticity. Results are summarized in Figure 6. Comparing Spectral 128 to FVM 1024 we can see that the spectral-only method, without any machine learning, already comes close to a similar improvement. Thus, a ~2x improvement over the spectral-only baseline — achieved by both the EPD and our hybrid model — is comparable to the ~8x improvement over FVM achieved in Kochkov et al. (2021) in which they used a hybrid ML-Physics model.

To understand the importance of data representation, we compare three representations for inputs to neural networks in our EPD and split-operator methods: velocity, vorticity, and velocity-vorticity concatenation. All models represent internal state with vorticity and learn corrections to vorticity, with the exception of velocity-only EPD models, which represent with and learn corrections to velocity to match Kochkov et al. (2021) and Stachenfeld et al. (2022). Interestingly, the velocity-only representation has the best performance across the board, particularly for the fully learned EPD model, as shown in the right panel of Figure 6. The nonlinear term $\mathbf{v} \cdot \nabla\omega$ in Equation (6) makes use of both vorticity and velocity representations. This indicates that in order to model the dynamics, the network must learn to solve for velocity — a global operation — making it challenging for a ConvNet with local convolutions to model. In contrast, computing vorticity from velocity only requires evaluating derivatives, which can be easily evaluated with local convolutions, e.g., via finite differences. This may explain the relatively-worse performance of Fourier Neural Operators using the velocity-based representation of Stachenfeld et al. (2022) versus the vorticity-based representation of Li et al. (2021).

To understand the value of time-splitting for the learned correction, we also performed an experiment where we instead incorporated our ML model into the nonlinear part of Physics-Step (Eq. (3)). This entails solving the convection term with the same 4th order explicit Runge-Kutta method used in the numerical solver. Since velocity-space performed best with the split operator model, we also used it here. The non-split learned correction has significantly worse performance. Anecdotally, we observed that training with high order Runge-Kutta methods was significantly less stable. Models were much less robust to small changes in learning rates and dilation rates (App. C) than models trained with first-order Runge-Kutta. We believe

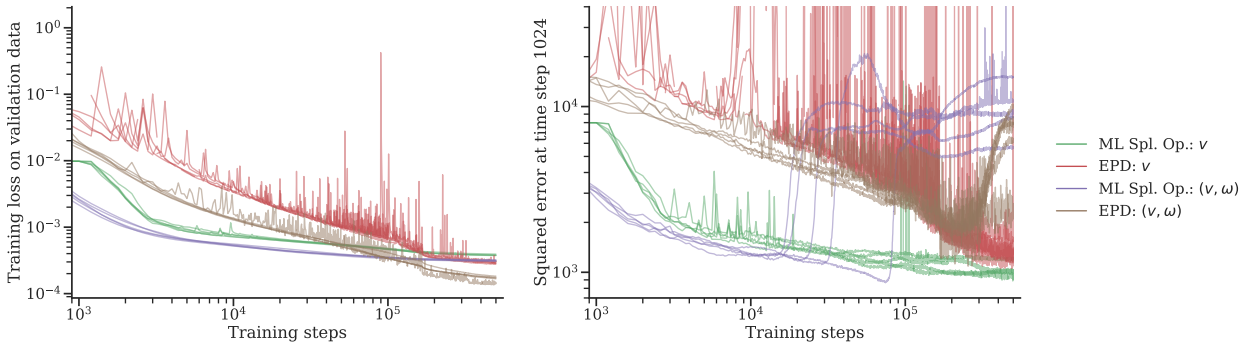

Figure 7: Learning curves for selected models, showing evaluation metrics after different numbers of training steps on the held-out validation dataset. The training loss is evaluated over the first 32 unrolled steps, whereas squared error is evaluated at 1024 steps ($t = 7.2$). The latter better indicates the model performance we care about for predicting long trajectories. Separate lines show learning curves for each of our five randomly initialized training runs.

this may be due to the increased difficulty of propagating gradients through a network which is effectively four times deeper. Another possibility is that the polynomial approximation of the high-order Runge-Kutta method introduces larger errors at the beginning of training, when the model is less accurate.

To further compare the merits of different ML approaches, we compare the learning curves for select models in Figure 7. First, we notice that although our best EPD and hybrid models are similarly accurate once trained, the hybrid models require drastically less compute to achieve fixed evaluation metrics. For example, squared error of 2000 at 1024 time-steps requires only 3600 training steps for the median ML split operator velocity model, versus 164 600 training steps for the EPD velocity model, corresponding to 4.4 versus 170 TPU-core hours.

These learning curves also reveal that our validation loss, in this case calculated over 32 unrolled time-steps, is not necessarily indicative of validation performance over much longer unrolls. Although our models have never seen the exact validation data during training, many of them are still able to "overfit" to the task of predicting short trajectories, at the cost of generalization to long unrolls. We also observe that different ML architectures overfit to different extents. In particular, pure ML models and models with access to vorticity overfit more than our best hybrid model, the ML split operator with only velocity inputs. The ML split operator with access to both velocity and vorticity is particularly interesting, because it seems to undergo a phase transition at around 30 000 training steps, with correlation for long unrolls dropping dramatically as the model shifts into the "memorization" regime.

5 Discussion

In this paper, we demonstrated the potential of ML-augmented spectral solvers to improve upon the accuracy of pure spectral methods. We also identified several key physically motivated modelling choices — velocity-representation, first-order time stepping to improve sensitivity to hyperparameters, and the removal of global spatial artifacts — which improve training for both ML-only and hybrid models. Pure ML models can match the accuracy of hybrid models, but are considerably more expensive to train and show more indications of overfitting.

Traditional spectral methods are powerful set of approaches for solving equations with smooth, periodic solutions. It is yet unclear whether ML-based solvers can achieve meaningful computational speed-ups over these strong baselines. In contrast to prior work which showed computational speed-ups of up to 1-2 orders of magnitude over baseline finite volume (Kochkov et al., 2021) and finite difference methods (List et al., 2022), the roughly 2× decrease in grid resolution for 2D turbulence with the ML split operator would only allow for at most 8× reduction in computational cost. However, our neural network for learned corrections is about 10× slower than Physics-Step, which entirely counteracts this potential gain. Small speed-ups

might be obtained by using smaller networks or applying corrections less frequently, but overall there is little potential for accelerating 2D turbulence beyond traditional spectral solvers.

Hybrid ML-spectral methods may enjoy more significant improvements on other problems. Examples might include PDEs with less smooth solutions, such as 3D turbulence, where energy decays as $k^{-5/3}$ vs k^{-3} in 2D (Pope, 2000), and global atmospheric models, where spectral methods do not achieve exponential convergence (Williamson, 2008). Cases where the exact governing equations are partially unknown, such as physical parameterizations for climate models (Brenowitz & Bretherton, 2018), also present an opportunity for hybridizing physical simulation with machine learning components.

References

Yohai Bar-Sinai, Stephan Hoyer, Jason Hickey, and Michael P. Brenner. Learning data-driven discretizations for partial differential equations. *Proceedings of the National Academy of Sciences*, 116(31):15344–15349, July 2019. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1814058116. URL https://pnas.org/doi/full/10.1073/pnas.1814058116. (↑ 2, 7, 8, 16)

Guido Boffetta and Robert E. Ecke. Two-Dimensional Turbulence. *Annual Review of Fluid Mechanics*, 44 (1):427–451, January 2012. ISSN 0066-4189, 1545-4479. doi: 10.1146/annurev-fluid-120710-101240. URL https://www.annualreviews.org/doi/10.1146/annurev-fluid-120710-101240. (↑ 8)

J P Boyd. The Erfc-Log filter and the asymptotics of the euler and vandeven sequence accelerations. In Ilin, A V and Scott, L Ridway (ed.), *Proceedings of the Third International Conference on Spectral and High Order Methods*, pp. 267–276, Houston, 1996. Houston Journal of Mathematics. (↑ 6)

John P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Books on Mathematics. Dover Publications, Mineola, NY, second edition, 2001. ISBN 0486411834 9780486411835. (↑ 2)

N. D. Brenowitz and C. S. Bretherton. Prognostic validation of a neural network unified physics parameterization. *Geophysical Research Letters*, 45(12):6289–6298, 2018. doi: https://doi.org/10.1029/2018GL078510. URL https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2018GL078510. (↑ 11)

Oscar P. Bruno, Jan S. Hesthaven, and Daniel V. Leibovici. FC-based shock-dynamics solver with neural-network localized artificial-viscosity assignment, 2021. URL https://arxiv.org/abs/2111.01315. (↑ 2)

Keaton J Burns, Geoffrey M Vasil, Jeffrey S Oishi, Daniel Lecoanet, and Benjamin P Brown. Dedalus: A flexible framework for numerical simulations with spectral methods. *Phys. Rev. Research*, 2(2):023068, April 2020. URL https://link.aps.org/doi/10.1103/PhysRevResearch.2.023068. (↑ 1)

C. Canuto, M.Y. Hussaini, A. Quarteroni, and T.A. Zang. *Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics*. Scientific Computation. Springer Berlin Heidelberg, 2007. ISBN 9783540307280. URL https://books.google.com/books?id=iDckv0W52cQC. (↑ 2, 15)

Claudio Canuto, M Yousuff Hussaini, Alfio Quarteroni, and Thomas A Zang. *Spectral methods: Fundamentals in single domains*. Scientific computation. Springer, Berlin, Germany, 1 edition, April 2006. URL http://link.springer.com/10.1007/978-3-540-30726-6. (↑ 6)

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf. (↑ 5)

J. W. Cooley and J. W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Math. Comput.*, 19(90):297–301, 1965. ISSN 00255718, 10886842. URL http://www.jstor.org/stable/2003354. (↑ 4)

Paul A Durbin. Some recent developments in turbulence closure modeling. *Annu. Rev. Fluid Mech.*, 50(1): 77–103, 2018. URL http://dx.doi.org/10.1146/annurev-fluid-122316-045020. (↑ 1)

Yuwei Fan, Jordi Feliu-Fabà, Lin Lin, Lexing Ying, and Leonardo Zepeda-Núñez. A multiscale neural network based on hierarchical nested bases. *Research in the Mathematical Sciences*, 6(2):21, Mar. 2019. ISSN 2197-9847. doi: 10.1007/s40687-019-0183-3. (↑ 2)

Hugo Frezat, Julien Le Sommer, Ronan Fablet, Guillaume Balarac, and Redouane Lguensat. A posteriori learning for quasi-geostrophic turbulence parametrization. *arXiv*, April 2022. URL http://arxiv.org/abs/2204.03911. (↑ 1, 2, 6)

D. Gottlieb and J. S. Hesthaven. Spectral methods for hyperbolic problems. *Journal of Computational and Applied Mathematics*, 128(1):83–131, March 2001. ISSN 0377-0427. doi: 10.1016/S0377-0427(00)00510-0. URL https://www.sciencedirect.com/science/article/pii/S0377042700005100. (ˆ 2, 6)

David Gottlieb and Steven A. Orszag. *Numerical Analysis of Spectral Methods*. Society for Industrial and Applied Mathematics, 1977. doi: 10.1137/1.9781611970425. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611970425. (ˆ 2)

Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proc. Natl. Acad. Sci. U. S. A.*, 118(21), May 2021. URL https://www.pnas.org/content/118/21/e2101784118. (ˆ 1, 2, 6, 7, 8, 9, 10, 16)

David A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 9048122600. (ˆ 2)

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Multipole graph neural operator for parametric partial differential equations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546. (ˆ 2)

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier Neural Operator for Parametric Partial Differential Equations. *arXiv:2010.08895 [cs, math]*, May 2021. URL http://arxiv.org/abs/2010.08895. arXiv: 2010.08895. (ˆ 1, 2, 9)

Björn List, Li-Wei Chen, and Nils Thuerey. Learned Turbulence Modelling with Differentiable Fluid Solvers. *arXiv:2202.06988 [physics]*, February 2022. URL http://arxiv.org/abs/2202.06988. arXiv: 2202.06988. (ˆ 1, 2, 10)

Robert I. McLachlan and G. Reinout W. Quispel. Splitting methods. *Acta Numerica*, 11:341–434, 2002. doi: 10.1017/S0962492902000053. (ˆ 5)

Siddhartha Mishra. A machine learning framework for data driven acceleration of computations of differential equations. *Mathematics in Engineering*, 1(1):118–146, 2019. ISSN 2640-3501. doi: 10.3934/Mine.2018.1.118. URL http://www.aimspress.com/rticle/doi/10.3934/Mine.2018.1.118. (ˆ 2)

Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, August 2000. ISBN 9780521598866. (ˆ 1, 2, 11)

Stephen B Pope. Ten questions concerning the large-eddy simulation of turbulent flows. *New Journal of Physics*, 6:35–35, March 2004. ISSN 1367-2630. doi: 10.1088/1367-2630/6/1/035. URL https://iopscience.iop.org/article/10.1088/1367-2630/6/1/035. (ˆ 1)

Christopher D Roberts, Retish Senan, Franco Molteni, Souhail Boussetta, Michael Mayer, and Sarah PE Keeley. Climate model configurations of the ECMWF integrated forecasting system (ECMWF-IFS cycle 43r1) for HighResMIP. *Geoscientific model development*, 11(9):3681–3712, 2018. URL https://doi.org/10.5194/gmd-11-3681-2018. (ˆ 1)

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pp. 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4. (ˆ 2)

Hidetsugu Sakaguchi. Chaotic dynamics of an unstable Burgers equation. *Physica D: Nonlinear Phenomena*, 129(1):57–67, May 1999. ISSN 0167-2789. doi: 10.1016/S0167-2789(98)00317-0. URL https://www.sciencedirect.com/science/article/pii/S0167278998003170. (ˆ 8, 15)

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to Simulate Complex Physics with Graph Networks. *arXiv:2002.09405 [physics, stat]*, September 2020. URL http://arxiv.org/abs/2002.09405. arXiv: 2002.09405. (ˆ 2)

Kimberly Stachenfeld, Drummond B. Fielding, Dmitrii Kochkov, Miles Cranmer, Tobias Pfaff, Jonathan Godwin, Can Cui, Shirley Ho, Peter Battaglia, and Alvaro Sanchez-Gonzalez. Learned Coarse Models for Efficient Turbulence Simulation. *arXiv:2112.15275 [physics]*, January 2022. URL http://arxiv.org/abs/2112.15275. arXiv: 2112.15275. (ˆ 1, 2, 5, 7, 9, 16)

Gilbert Strang. On the construction and comparison of difference schemes. *SIAM Journal on Numerical Analysis*, 5(3):506–517, 1968. doi: 10.1137/0705041. URL https://doi.org/10.1137/0705041. (ˆ 5)

Eitan Tadmor. The exponential accuracy of Fourier and Chebyshev differencing methods. *SIAM Journal on Numerical Analysis*, 23(1):1–10, 1986. doi: 10.1137/0723001. URL https://doi.org/10.1137/0723001. (ˆ 4)

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized Fourier Neural Operators. *arXiv:2111.13802 [cs]*, November 2021. URL http://arxiv.org/abs/2111.13802. arXiv: 2111.13802. (ˆ 2)

Lloyd N. Trefethen. *Spectral Methods in MATLAB*. Society for industrial and applied mathematics (SIAM), 2000. URL https://doi.org/10.1137/1.9780898719598. (ˆ 1, 2, 4)

Kiwon Um, Raymond, Fei, Philipp Holl, Robert Brand, and Nils Thuerey. Solver-in-the-Loop: Learning from differentiable physics to interact with iterative PDE-Solvers. In *NeurIPS 2020*, June 2020. URL http://arxiv.org/abs/2007.00016. (ˆ 6, 8)

Rui Wang, Karthik Kashinath, Mustafa Mustafa, Adrian Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020. URL http://dx.doi.org/10.1145/3394486.3403198. (ˆ 2)

David L Williamson. Equivalent finite volume and eulerian spectral transform horizontal resolutions established from aqua-planet simulations. *Tellus Ser. A Dyn. Meteorol. Oceanogr.*, 60(5):839–847, January 2008. URL https://doi.org/10.1111/j.1600-0870.2008.00340.x. (ˆ 11)

P K Yeung and K Ravikumar. Advancing understanding of turbulence through extreme-scale computation: Intermittency and simulations at large problem sizes. *Phys. Rev. Fluids*, 5(11):110517, November 2020. URL https://link.aps.org/doi/10.1103/PhysRevFluids.5.110517. (ˆ 4)

Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In Yoshua Bengio and Yann LeCun (eds.), *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL http://arxiv.org/abs/1511.07122. (ˆ 17)