Learning to correct spectral methods for simulating turbulent flows

Gideon Dresdner *gideond@gmail.com*
Google Research and ETH Zurich, Department for Computer Science

Dmitrii Kochkov *dkochkov@google.com*
Google Research

Peter Norgaard *pnorgaard@google.com*
Google Research

Leonardo Zepeda-Núñez *lzepedanunez@google.com*
Google Research

Jamie A. Smith *jamieas@google.com*
Google Research

Michael P. Brenner *mbrenner@google.com*
Google Research

Stephan Hoyer *shoyer@google.com*
Google Research

Reviewed on OpenReview: *https://openreview.net/forum?id=wNBARGxoJn*

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 and 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 known to be more efficient than other numerical schemes for simulating PDEs with smooth and periodic solutions. Specifically, we develop ML-augmented spectral solvers for three common PDEs of fluid dynamics. Our models are more accurate ($2-4\times$) than standard spectral solvers at the same resolution but have longer overall runtimes ($\sim 2\times$), due to the additional runtime cost of the neural network component. 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 infeasible 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 are 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). Going beyond accuracy, hybrid methods have combined classical numerical simulation with ML to improve stability and generalize to new physical systems, e.g., out of sample distributions with different forcing setups (Kochkov et al., 2021; List et al., 2022).

However, hybrid methods have 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 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 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 provide some improvement on the accuracy of spectral-only methods, which themselves perform remarkably well compared to recently proposed ML methods. 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 an emerging set of approaches which aim to combine classical numerical methods with contemporary data-driven deep learning techniques. These approaches use high-resolution simulation data to learn corrections to low-resolution numerical schemes with the goal of combining the best of both worlds — the simplicity of PDE-based governing equations and the expressive power of neural networks.

Our work falls into this category, along with a growing body of work (Mishra, 2018; Bar-Sinai et al., 2019; Kochkov et al., 2021; Bruno et al., 2021; List et al., 2022; Frezat et al., 2022; Huang et al., 2022).

Some of these works stand out as being closely related to ours but with important differences. Huang et al. (2022) also develops a hybrid approach but focuses exclusively on ODEs, not PDEs, and it is not clear whether their techniques for stable simulations with large time-step size would also apply to PDEs. In their recent work, Frezat et al. (2022) augmented spectral solvers for closure models exclusively focused on climate modeling. We also learn corrections to spectral methods but rather than focusing on climate models, we tackle more general PDEs, provide key ML architecture choices, and include comparisons to classical numerical methods on multiple grid resolutions.

Unlike our work, San & Maulik (2018) and Subel et al. (2021) take fundamentally different approaches to ours. San & Maulik (2018) improves classical approaches for closure modeling by using a neural network with a single hidden layer to learn fine-tuned local corrections instead of the usual global correction methods. Their work requires specialized knowledge of classical methods (POD, closure models, etc.) which is orthogonal to our larger goal of automating these approaches using data-driven techniques. Subel et al. (2021) focus on a data augmentation scheme to overcome stationary shocks that arise in Burgers' equation. Their proposed shifting mechanism could be used in conjunction with our work, but represents a different research direction.

2 Spectral methods for fluids

Spectral methods are a powerful method for finding high-accuracy solutions to PDEs and 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). Below, we provide a succinct introduction.

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, 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 maintain an accurate solution. Thus, runtime for PDE solvers is $O(n^{d+1})$, where d is the number of spatial dimensions and n is the number of discretization points per 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 stronger than diffusion.

For most turbulent flows of interest, closely approximating the exact PDE (known as Direct Numerical Simulation) is computationally intractable because it requires prohibitively high 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 determined by a closure model to account for averaged effects over fine spatial length scales. In practice, this term is often omitted due to the difficulty of deriving appropriate closure formulas ("implicit LES") and the PDE is simply simulated with the finest computationally feasible grid resolution. 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.

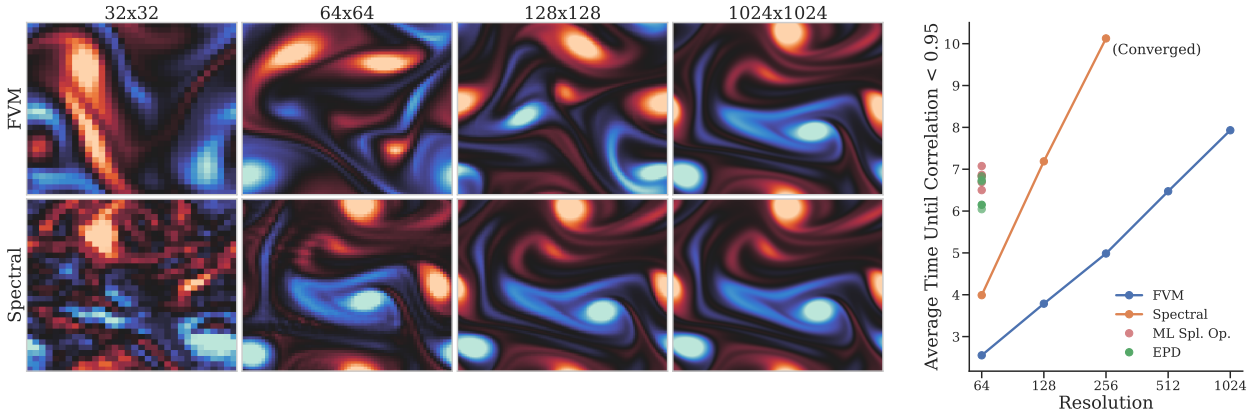

Figure 1: Comparing finite volume (FVM) to spectral convergence for 2D turbulence. *Left*: Vorticity fields evolved up to time 9.0 with different grid sizes and numerical methods, starting from an identical initial condition. Qualitatively, it is clear that at resolution 64x64, the spectral method has 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 the EPD baseline and our ML Split Operator (ML Spl. Op.) models, we show performance across five different neural network parameter initializations.

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 2π-periodic, square-integrable for all times t, and for simplicity, one-dimensional. Consider the Fourier coefficients $\widehat{\mathbf{u}}^t$ of $\mathbf{u}(x,t)$, truncated to lowest $K+1$ frequencies (for even K):

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

$\widehat{\mathbf{u}}^t$ 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 $\widehat{\mathbf{u}}^t_k$ 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, in turn, 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 $\widehat{\mathbf{u}}^t_k$ 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 (i.e., faster than any power of $1/K$) 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.

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

3 Learned split operators for correcting spectral methods

Similarly to classical spectral 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 spectral method, denoted Physics-Step. The machine learning component, denoted Learned-Correction$(\cdot\,;\theta)$, contains tune-able parameters θ which are optimized during training to match high-resolution simulations.

Building upon traditional physics-based solvers, we use a simple explicit-Euler time integrator for the correction term (See App. D for a brief explanation of explicit methods.) This yields the following update equation:

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

where $h \in \mathbb{R}$ is the time step size. Ultimately, our work is aimed at two and three dimensional problems so rather than present an abstract schematic as presented in the equation above (Eq. (3)), we chose to present a more detailed, realistic schema of 2D turbulence (Eq. (6)). The spatial state variable is thus not the generalized $\mathbf{u}$ as in Equation (3) but rather, the vorticity ω as defined in Section 4.5

3.1 Convolutional layers for encoding the physical prior of locality

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. We now provide a high-level view of our modeling choices:

Real-space vs. frequency-space. 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 inverse-FFT and FFT to map the signal back and forth between frequency- and real-space.

ConvNet Padding. In this work, we tackle problems with periodic boundary conditions. This makes periodic padding a natural choice for the ConvNets.

Neural architecture. For both 1D and 2D problems we used an Encoder-Process-Decoder (EPD) architecture (Stachenfeld et al., 2022) to facilitate fairer comparisons to pure neural network baselines. While various forms of optimization is possible, such as model parallelism among others, we consider this to be outside the scope of this exploratory work. See Section 4.2 for a detailed description of EPD models and Appendix C for further information.

3.2 The split operator method for combining time scales

Due to a variety of considerations — numerical stability, computational feasibility, etc. — each term of a PDE often warrants its own time-advancing method. This motivates *split operator methods* (Strang, 1968; McLachlan & Quispel, 2002), a popular tool for solving PDEs which combine different time integrators. In this work, we use the split operator approach to incorporate the additional terms given by the neural network.

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 a 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 when compared to incorporating the learned component in the 4th-order Runge-Kutta solver, is more accurate and has 4x faster runtime (see Sec. 4). Alternatively, omitting Physics-Step from Equation (3) gives

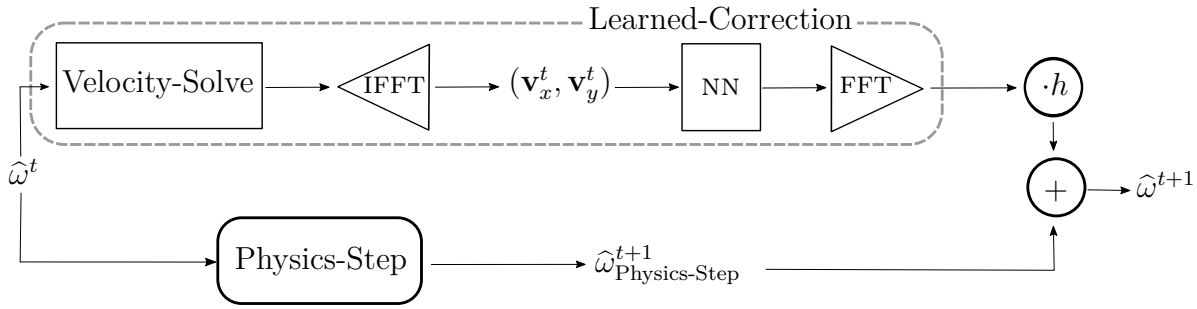

Figure 2: Diagram summarizing our model described in Equation (3) for the 2D Navier-Stokes equation. The input, vorticity $\widehat{\omega}^t$, is processed by two independent components — Learned-Correction and Physics-Step — operating at different time-scales. The output of Learned-Correction is weighted 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.

a Neural ODE model (Chen et al., 2018) with first-order time-stepping which is precisely the EPD model described by Stachenfeld et al. (2022). This model serves as a strong baseline as shown 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. We found that ML models operate better in velocity-space whereas vorticity is the more suitable representation for the numerical solver. We were able to improve accuracy by incorporating a physics-based data pre-processing step, e.g., a velocity-solve operation, for the inputs of the neural network component (See App. D for a brief explanation of velocity-solve). Our overall model is Learned-Correction = $\text{FFT}(\text{NN}(\text{IFFT}(\text{State-Transform}(\widehat{\mathbf{u}}^t))))$, where 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. 2) we perform a velocity-solve operation 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 this is because insufficiently filtered simulations will often entirely diverge rather than accumulate small errors. Thus, in addition to filtering the downsampled training data, we also used the exponential filter on the outputs of Physics-Step of Equation (3) and consider this to be another component of Physics-Step. Omitting this filtering also resulted in global errors which were impossible for the learned component to correct. Figure 3 summarizes our data generation pipeline, including an explicit filtering step. In Section 4, we present a failure mode on a model without filtering (see Fig. 5).

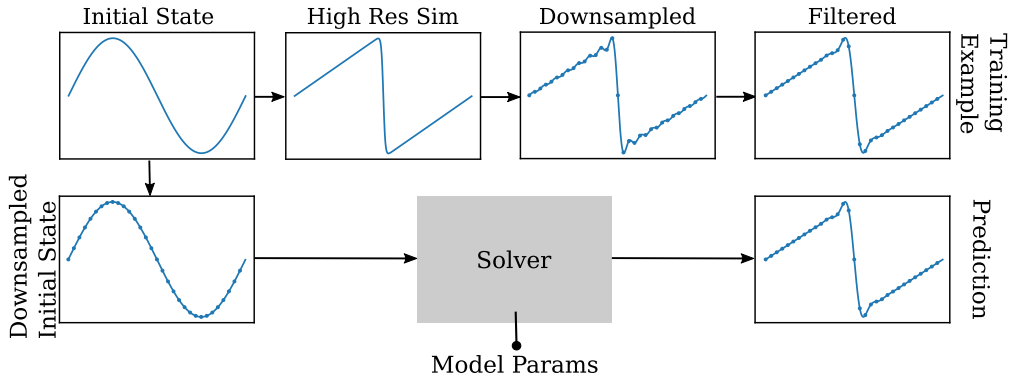

Figure 3: Diagram of our training pipeline. Starting with a high resolution initial state, we run it forward using a high-resolution spectral solver. Then we downsample by truncating higher frequencies in the Fourier representation. This can cause "ringing effects" for which the standard approach is to apply a filtering operator. Finally, we train a solver to mimic this process as closely as possible, as measured in ℓ_2-loss.

We experimented with larger time-step sizes, e.g. $h = 0.1$, $h = 1.0$ but found the resulting models to be unstable. This is because the physics-component of the model quickly becomes unstable at large step sizes which propagates to the overall model.

Training loss. We train our models to minimize the squared-error over some number of unrolled prediction steps, which allows our model to account for compounding errors (Um et al., 2020). Let $\overline{\mathbf{u}}(x,t)$ denote our prediction at time t, then our training objective is given by $\beta \sum_{x,t \leq T} |\overline{\mathbf{u}}(x,t) - \mathbf{u}(x,t)|^2$. The scaling constant β is chosen so that the loss of predicting "no change" is one, i.e., $\beta^{-1} = \sum_{x,t \leq T} |\mathbf{u}(x,0) - \mathbf{u}(x,t)|^2$. Relative to the thousands of time-steps over which we hope to simulate accurately, we unroll over a relatively small number during training (e.g., $T = 32$ for 2D turbulence) because training over long time windows is less efficient and less stable (Kochkov et al., 2021).

Measuring convergence. We seek to optimize the accuracy of coarse-grained simulations. More specifically, 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 $\text{Corr}[\mathbf{u}(\cdot,t), \overline{\mathbf{u}}(\cdot,t)] = \sum_x (\overline{\mathbf{u}}(x,t) \cdot \mathbf{u}(x,t)) / (\|\overline{\mathbf{u}}(x,t)\|_2 \|\mathbf{u}(x,t)\|_2)$ (since our flows have mean zero). Finally, we compute the first time step in which the correlation dips below 0.95, i.e. $\min\{t \mid \text{Corr}[\mathbf{u}(\cdot,t), \overline{\mathbf{u}}(\cdot,t)] < 0.95\}$. While MAE is precise up to floating point precision, it is not readily interpretable. Whereas, correlation is less sensitive but more interpretable. For this reason, we prefer to report correlation. This potential redundancy is demonstrated in our experiment on the KS equation (Sec. 4.3), where we included both MAE and correlation.

For the 2D Navier-Stokes equation, we only report correlation because it is sufficient. For the unstable Burgers' equation, the situation was the opposite: We only reported MAE because measuring correlation did not provide additional insight — all models had correlation values close to 1.0, whereas MAE was sensitive to the improved performance of our method.

4 Results

4.1 Model equations

We showcase our method using three model equations which capture many of the algorithmic difficulties present in more complex systems: 1D Kuramoto–Sivashinsky (KS) equation, 1D unstable Burgers' equation, and 2D 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. Therefore, it is not surprising that, while our method does improve over spectral-only methods, that improvement is not significant. On the other hand, the unstable Burgers' equation presents a test case in which classical spectral methods struggle near discontinuities. Here, our method outperforms spectral-only methods. Finally, with two-dimensional Kolmogorov flow, we demonstrate our method on a more challenging fluid simulation. Kolmogorov flow exhibits multiscale behavior in addition to smooth, chaotic dynamics. Without any modification, spectral-only methods are already competitive with the latest hybrid methods (Kochkov et al., 2021). Similar to the 1D KS equation, our method is able to provide some improvement in this case.

4.2 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 learned component of 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 this FVM model as a baseline.

We avoided extensive comparisons to classical subgrid modeling, such as Large Eddy Simulation (LES), since LES is itself a nuanced class of methods with many tunable parameters. Instead, our physics-only baselines are implicit LES models — a widely-used and well-understood model class which serves as a consistent, parameter-free baseline. In contrast, explicit sub-grid-scale models such as Smagorinsky models include tunable parameters, with the optimal choice depending on the scenario of interest. Furthermore, explicit LES models typically focus on matching the energy spectrum rather than minimizing point-wise errors (List et al., 2022).

4.3 Kuramoto-Sivashinsky (KS) equation

The Kuramoto-Sivashinsky (KS) equation is a model of unstable flame fronts. In the PDE literature, it is a popular model system because it exhibits chaotic dynamics in only a single dimension. 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 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.

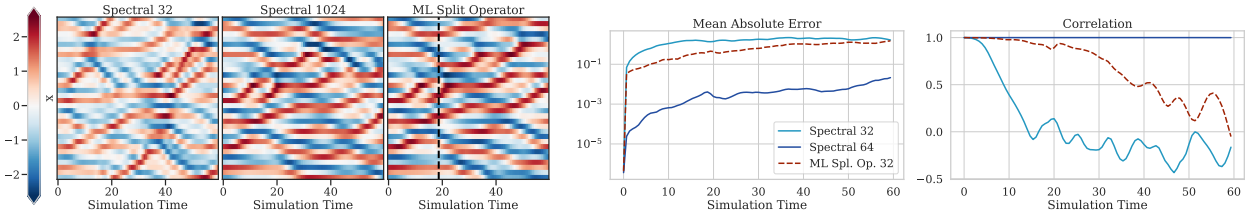

Figure 4: Comparing our model to spectral-only baselines on the KS equation. The spectral method is essentially converged to a 1024 ground truth model at a resolution of 64. Vertical dashed line indicates the first time step in which our model's correlation with the ground truth is less than 0.9. Our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

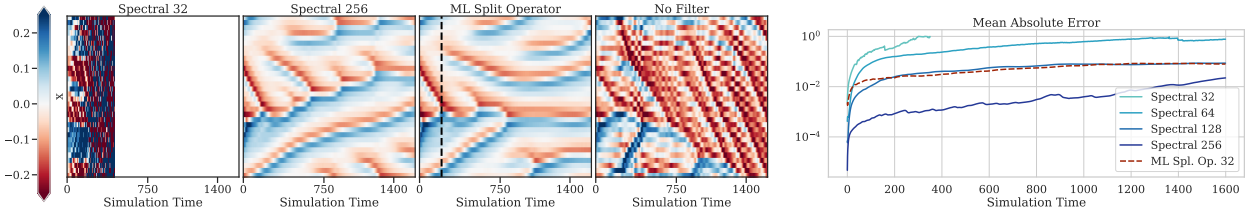

Figure 5: Comparing our model to spectral-only baselines on the unstable Burgers' equation (Eq. (5)). Vertical dashed line indicates the first time step in which our model's correlation with the ground truth is less than 0.9. Not only is our model stable unlike the coarse (Spectral 32) baseline, it also 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").

4.4 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 challenging to model using the Fourier basis because there are discontinuities in the 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 often results in instability at low resolutions, evidenced in our results (Fig. 5, resolution 32). While there are classical methods for addressing discontinuities, e.g., adding a hyper-viscosity term, they often modify the underlying dynamics that they are trying to model. The inadequacy of unaltered spectral methods for solving this problem explains why our method is able to achieve approximately 4x improvement in spatial resolution over a spectral baseline. Results are summarized in Figure 5.

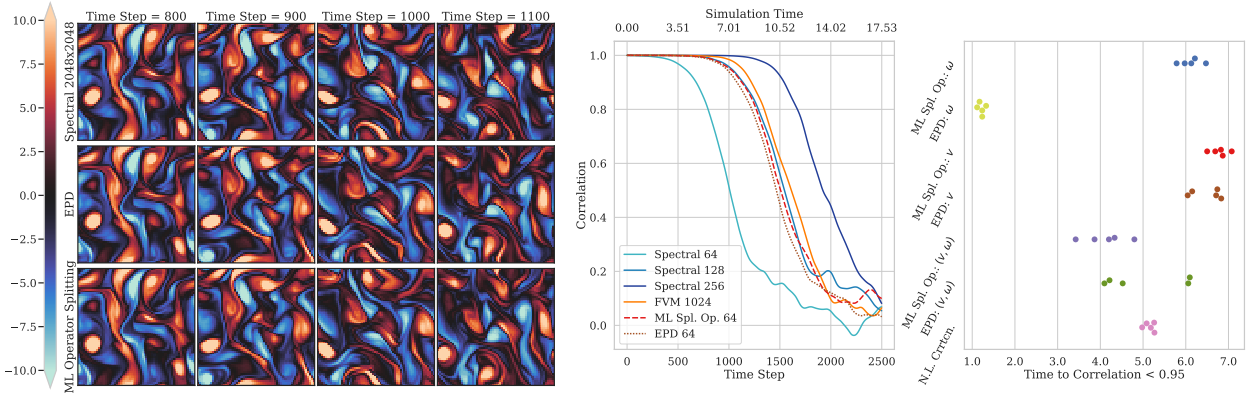

Figure 6: Benchmarking our Neural Split Operator (ML Spl. Op.) to spectral-only baselines and state-of-the-art ML-only, EPD model (Stachenfeld et al., 2022). *Left*: Visualizing model states starting at the time when divergence begins to occur. At time step = 800, EPD, Neural Split Operator model, and 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.

4.5 2D turbulence

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 vector field $\mathbf{v}(x, y) = (\mathbf{v}_x, \mathbf{v}_y)$ 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 can be recovered from the vorticity field by solving a Poisson equation $-\nabla^2 \psi = \omega$ for the stream function ψ and then using the relation $\mathbf{v}(x, y) = (\partial_y \psi, -\partial_x \psi)$. This velocity-solve operation is computed via element-wise multiplication and division in the Fourier basis.

For this example, 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 use vorticity as their state representation. The nonlinear term correction method uses a neural network to correct the inputs to the nonlinear term $\mathbf{v} \cdot \nabla \omega$ and avoids using classical correction techniques, e.g., the so-called three-over-two rule (Orszag, 1971). Specifically, this approach takes as inputs $(\mathbf{v}, \nabla \omega)$ and outputs a scalar field which is incorporating into the explicit part of the physics solver, e.g. $\partial_t \omega = -(\mathbf{v} \cdot \nabla \omega + \text{Nonlinear-Correction}(\mathbf{v}, \omega; \theta)) + \eta \nabla^2 \omega - \alpha \omega + f$

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.

State 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. These models never compute vorticity — they use velocity to represent state as well as learned corrections, thus matching previous work by Kochkov et al. (2021) and Stachenfeld et al. (2022). Interestingly, this velocity-only representation has the best performance across the board, particularly for the fully learned EPD model, even outperforming velocity-vorticity concatenation (Fig. 6, right panel).

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 restricted to local convolutions to learn. 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).

Full versus nonlinear-term-only correction. 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 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.

Analyzing learning curves. 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 spectral-only methods. We also identified several key physically motivated modeling 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.

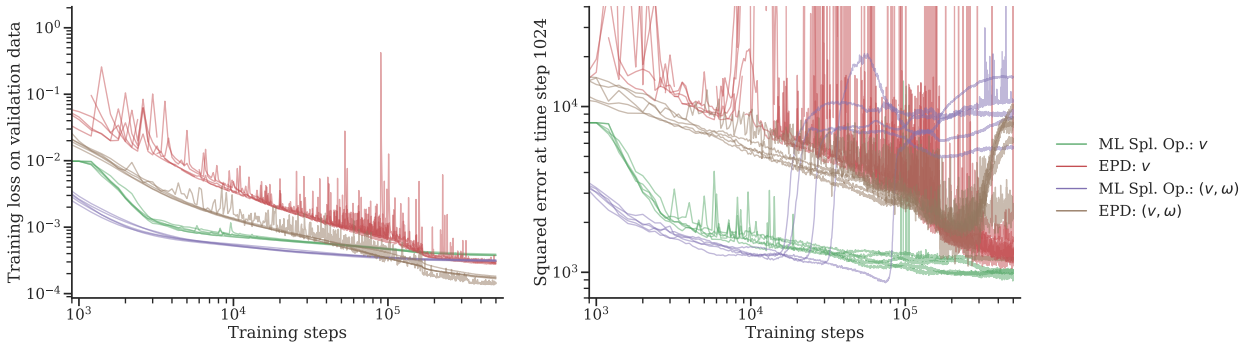

Figure 7: Learning curves for selected models, showing evaluation metrics after different numbers of training steps on the held-out validation dataset. The training loss is evaluated over the first 32 unrolled steps, whereas squared error is evaluated at 1024 steps ($t = 7.2$). The latter better indicates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

Traditional spectral methods are a 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 classical spectral methods on PDEs of this class. 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 allow for at most 8× reduction in computational cost. However, our neural network for learned corrections is about 10× slower than Physics-Step, which 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 smooth, periodic 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 $k^{-5/3}$ versus k^{-3} in 2D (Pope, 2000). Or, 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 combining 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, July 2019. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1814058116. (^ 3, 8, 9, 17)

Guido Boffetta and Robert E. Ecke. Two-Dimensional Turbulence. *Annual Review of Fluid Mechanics*, 44, January 2012. ISSN 0066-4189, 1545-4479. doi: 10.1146/annurev-fluid-120710-101240. (^ 10)

John 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*, 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. (^ 3)

N. D. Brenowitz and C. S. Bretherton. Prognostic Validation of a Neural Network Unified Physics Parameterization. *Geophysical Research Letters*, 45, 2018. doi: https://doi.org/10.1029/2018GL078510. (^ 12)

Oscar P. Bruno, Jan S. Hesthaven, and Daniel V. Leibovici. FC-based shock-dynamics solver with neural-network localized artificial-viscosity assignment, 2021. (^ 3)

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, April 2020. (^ 2)

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. (^ 3, 16)

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. (^ 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. (^ 6)

J. W. Cooley and J. W. Tukey. An Algorithm for the Machine Calculation of Complex Fourier Series. *Math. Comput.*, 19, 1965. ISSN 00255718, 10886842. (^ 4)

Paul A Durbin. Some Recent Developments in Turbulence Closure Modeling. *Annu. Rev. Fluid Mech.*, 50, 2018. (^ 2)

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, 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. (^ 2, 3, 6)

D. Gottlieb and J. S. Hesthaven. Spectral methods for hyperbolic problems. *Journal of Computational and Applied Mathematics*, 128, March 2001. ISSN 0377-0427. doi: 10.1016/S0377-0427(00)00510-0. (^ 3, 6)

David Gottlieb and Steven A. Orszag. *Numerical Analysis of Spectral Methods*. Society for Industrial and Applied Mathematics, 1977. doi: 10.1137/1.9781611970425. (^ 3)

Zhongzhan Huang, Senwei Liang, Hong Zhang, Haizhao Yang, and Liang Lin. Accelerating Numerical Solvers for Large-Scale Simulation of Dynamical System via NeurVec, August 2022. arXiv:2208.03680. (^ 3)

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, May 2021. (^ 2, 3, 7, 8, 10, 11, 12, 17)

David A. Kopriva. *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers*. Springer Publishing Company, Incorporated, 1st edition, 2009. ISBN 9048122600. (^ 3)

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, 2020. (^ 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. arXiv: 2010.08895. (^ 2, 11)

Björn List, Li-Wei Chen, and Nils Thuerey. Learned Turbulence Modelling with Differentiable Fluid Solvers. *arXiv:2202.06988 [physics]*, February 2022. arXiv: 2202.06988. (^ 2, 3, 8, 12)

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

Siddhartha Mishra. A machine learning framework for data driven acceleration of computations of differential equations. *Mathematics in Engineering*, 1, 2018. ISSN 2640-3501. doi: 10.3934/Mine.2018.1.118. (^ 3)

Steven A. Orszag. On the Elimination of Aliasing in Finite-Difference Schemes by Filtering High-Wavenumber Components. *Journal of the Atmospheric Sciences*, 28, September 1971. ISSN 0022-4928, 1520-0469. doi: 10.1175/1520-0469(1971)028<1074:OTEOAI>2.0.CO;2. (^ 10)

Stephen B. Pope. *Turbulent Flows*. Cambridge University Press, August 2000. ISBN 9780521598866. (^ 2, 12)

Stephen B Pope. Ten questions concerning the large-eddy simulation of turbulent flows. *New Journal of Physics*, 6, March 2004. ISSN 1367-2630. doi: 10.1088/1367-2630/6/1/035. (^ 2)

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, 2018. (^ 2)

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*, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4. (^ 2)

Hidetsugu Sakaguchi. Chaotic dynamics of an unstable Burgers equation. *Physica D: Nonlinear Phenomena*, 129, May 1999. ISSN 0167-2789. doi: 10.1016/S0167-2789(98)00317-0. (^ 9, 16)

Omer San and Romit Maulik. Extreme learning machine for reduced order modeling of turbulent geophysical flows. *Physical Review E*, 97(4):042322, April 2018. doi: 10.1103/PhysRevE.97.042322. Publisher: American Physical Society. (^ 3)

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. arXiv: 2002.09405. (^ 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. arXiv: 2112.15275. (^ 2, 5, 6, 8, 10, 11, 17)

Gilbert Strang. On the Construction and Comparison of Difference Schemes. *SIAM Journal on Numerical Analysis*, 5, 1968. doi: 10.1137/0705041. (^ 5)

Adam Subel, Ashesh Chattopadhyay, Yifei Guan, and Pedram Hassanzadeh. Data-driven subgrid-scale modeling of forced Burgers turbulence using deep learning with generalization to higher Reynolds numbers via transfer learning. *Physics of Fluids*, 33(3):031702, March 2021. ISSN 1070-6631, 1089-7666. doi: 10.1063/5.0040286. arXiv:2012.06664 [physics]. (^ 3)

Eitan Tadmor. The Exponential Accuracy of Fourier and Chebyshev Differencing Methods. *SIAM Journal on Numerical Analysis*, 23, 1986. doi: 10.1137/0723001. (^ 4)

Alasdair Tran, Alexander Mathews, Lexing Xie, and Cheng Soon Ong. Factorized Fourier Neural Operators. *arXiv:2111.13802 [cs]*, November 2021. arXiv: 2111.13802. (^ 2)

Lloyd N. Trefethen. *Spectral Methods in MATLAB*. Society for industrial and applied mathematics (SIAM), 2000. (^ 2, 3, 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. (^ 7, 9)

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. (^ 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, January 2008. (^ 12)

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, November 2020. (^ 4)

Z. Yin, H.J.H. Clercx, and D.C. Montgomery. An easily implemented task-based parallel scheme for the Fourier pseudospectral solver applied to 2D Navier–Stokes turbulence. *Computers & Fluids*, 33(4):509–520, 2004. ISSN 0045-7930. doi: 10.1016/j.compfluid.2003.06.003. (^ 19)

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. (^ 18)

Appendix

A Unstable Burgers

For convenience to the reader, we now provide the details of this equation as described in Sakaguchi (1999). Writing Equation (5) in Fourier space yields a convenient representation. Consider the k-th wavenumber:

$$\partial_t \widehat{\mathbf{u}}_k = -(\widehat{g}(k) + \nu)k^2 \widehat{\mathbf{u}}_k + \widehat{\mathbf{f}}_k \tag{7}$$

where $\widehat{\mathbf{f}}_k$ represents the contribution of the nonlinear term and $\widehat{g}$ is the Fourier transform of the convolution g. Now we can simply let $\widehat{g}(k) = -.04e^{-16k^2}$. (Note: In Sakaguchi (1999), the authors simply define g is Fourier space).

As shown in Figure 8, this definition for g amplifies low wave numbers while damping high wave numbers allowing for more complex dynamics than the original Burgers' Equation.

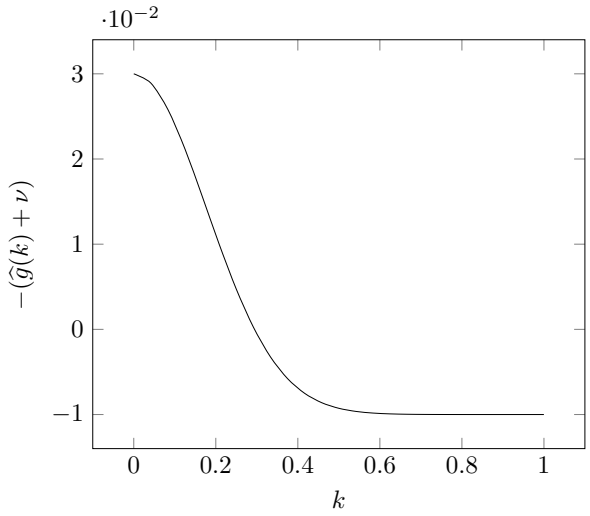

Figure 8: Plot of the scaling factor of the diffusion term in the Unstable Bergers' equation for each wave number k.

B Data generation

All training data was generated using pseudospectral solvers. We split the linear and nonlinear terms of the equation into implicit and explicit terms respectively, and used a Crank-Nicolson time stepping scheme with low storage fourth order Runge-Kutta, as described in Appendix D.3 of Canuto et al. (2007). Time step sizes were chosen according to the Courant–Friedrichs–Lewy (CFD) condition on the simulation grid, and are proportional increased for coarser simulations. Downsampling was then performed in space (Sec. 3.4) and in time. High resolution trajectories were downsampled to the target resolution (e.g. $N = 32$ or $N = 64$ grid points) using a block filter in Fourier space. Gibbs Phenomena were then removed from the downsampled trajectories using a low-pass filter as described in Section 3.4.

For simplicity, in Section 2.2 we assumed that the spatial domain of $\mathbf{u}$ is $[0, 2\pi]$. It is often convenient to regard the KS Equation and unstable Burgers' as having a larger domain since the domain size dictates the degree of chaos.

We generated data according to the following parameters:

Kuramoto-Sivashinsky

- Spatial domain: $[0, 64]$, to match Bar-Sinai et al. (2019)
- Number of samples: 16
- Warm-up time: 800.0
- Simulation time: 800.0
- Reference simulation grid: 1024
- Reference simulation time step: 0.020 833 3
- Viscosity: $\nu = 0.01$

Unstable Burgers'

- Spatial domain: $[0, 40\pi]$
- Number of samples: 16
- Warm-up time: 613.592
- Simulation time: 50 000
- Reference simulation grid: 1024
- Reference simulation time step: 0.061 359 2
- Viscosity: $\nu = 0.01$

2D Turbulence (Kolmogorov Flow)

- Spatial domain: $[0, 2\pi] \times [0, 2\pi]$
- Number of samples: 16
- Warm-up time: 40.0
- Simulation time: 30.0
- Reference simulation grid: 2048×2048
- Reference simulation time step: 0.000 219 140 11
- Viscosity: $\nu = 1.0 \times 10^{-3}$
- Drag coefficient: $\alpha = 0.1$
- Constant forcing $f((x, y)) = -k \cos(ky)$ with $k = 4$, which matches the velocity forcing from Kochkov et al. (2021) acting on vorticity.

For each set of parameters, we created a training and a validation dataset, which only differ in the random seed used to generate the initial conditions. All evaluation metrics are reported on the held-out validation data. By *sample* we mean a sample from the initial conditions, thus 16 samples refers to a dataset which was constructed from 16 distinct initial conditions.

C Learned model configurations

We followed the general architecture for Encoder-Process-Decoder models described in Stachenfeld et al. (2022) but with different hyperparameters for 1D and 2D equations.

Encoder and Decoder modules. The Encoder and Decoders modules consisted of a single convolutional layer with kernel size = 5. For 1D models we used 128 channels and for 2D models we used 64 channels.

Process module. The Process module consistent of residual blocks that uses dilated convolutions (Yu & Koltun, 2016), interspersed with ReLU nonlinearities. Dilation rates are written as $(1, 2)$, which denotes a two-layer network whose first layer has a dilation rate of one (e.g. no dilation), and second layer has dilation rate of two. For 1D models we used three residual blocks and for 2D Kolmogorov flow we used two blocks.

For 1D models the Process module had three layers with kernel size = 3 and no dilation, i.e. with rates $(1, 1, 1)$.

For Kolmogorov flow, we also used kernel size = 3. We experimented with the following dilation rates: $(1, 2)$ and $(1, 2, 4, 2, 1)$. For each of these, we tried the following learning rates: $[1\text{e}{-5}, 5\text{e}{-4}, 2.5\text{e}{-4}, 1.0\text{e}{-4}, 5.0\text{e}{-3}, 1.0\text{e}{-3}]$. While different models did seem to prefer certain dilation configurations, there was no general pattern across all models. For example, EPD with $(1, 2)$ had the best performance whereas for our split operator method, $(1, 2, 4, 2, 1)$ had better performance.

Neural network parameter initialization. We also tested for the sensitivity to neural network parameter initialization shown in Figure 1, *Right*. From the dilation configuration by learning rate sweep described above, we selected the best hyperparameter setting then tried 5 different random seeds for the neural network parameter initialization. Layer biases were initialized to zero and layer weights were initialized by truncated normal distributions.

Learning rates for 1D models. Generally, we found that performance was not significantly affected by learning rate, once a stable learning rate was found.

Number of unroll steps. The number of unroll steps for Kolmogorov flow was set to 32, and for both KS equation 8 and unstable Burgers' it was set to 8.

Optimizer. All machine learning models were trained using the Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.99$ and $\epsilon = 1\text{e}{-8}$. Our 2D turbulence models use a global batch size of 64 and train for 5e5 optimization steps. Each model takes approximately 65 wall-clock hours to train distributed across 8 TPU v4 cores.

Final model. Weights used for validation, including the learning curves plotted in Figure 7, use an exponential moving average of weights from training, with decay constant of 0.98.

Input/output scaling. Models which use ℓ_2-loss generally make a normality assumption about error distribution. When this assumption is violated, the gradient of the parameters is often too large at the beginning of learning resulting in unstable trajectories which are impossible to recover from. To address this, we observed the scale of the errors with an untrained model and then rescaled the outputs of the Learned-Correction component so that the errors had variance one. Specifically, we used the following scales:

- KS equation: 0.5
- Unstable Burgers': 0.1
- Kolmogorov Flow: 0.01

We found that for Kolmogorov Flow, it also helped to scale the inputs to the Learned-Correction term. We used a scale of 0.2 for all models on Kolmogorov Flow.

D Background: Computational Fluid Dynamics

Velocity-solve operation. By velocity-solve we mean a routine for mapping vorticity to velocity. Roughly speaking, this involves solving for the stream function and then uses the stream function to compute the

velocity which is the standard approach. We do this in the frequency domain by inverting the Laplace operator and scaling by the frequency mesh in the x and y-directions appropriately. A quick sketch can be found in Yin et al. (2004).

Courant–Friedrichs–Lewy (CFL) condition. The CFL condition dictates the required time-step size given a spatial step-size to ensure stability of numerical solutions to PDEs. Informally, the time-step size cannot exceed the maximum rate of change (velocity) of the system. Thus,

$$\frac{\Delta x}{\Delta t} \leq C v_{\max} \tag{8}$$

where Δx is given by the desired spatial resolution, the maximum velocity $v_{\max}$ is computed empirically by running progressively finer simulation until convergence, and C often depends on the type of time-stepping scheme.

Implicit and explicit time integration schemes. Explicit refers to the usual forward Euler integration scheme whereas implicit refers to the backward Euler integration scheme. The backward method, while more stable, often involves solving an equation algebraically to obtain an update step, e.g. u_{t+1} in terms of u_t – thus the term implicit since it defines an update step only implicitly. Both methods are standard practice in any form of numerical simulation, and the choice of either type of method depends on the properties of the equation being solved.