

Solver-in-the-loop approach to closure of shell models of turbulence

André Freitas ^{1,2,*}, Kiwon Um ², Mathieu Desbrun ³, Michele Buzzicotti ¹,
and Luca Biferale¹

¹*Department of Physics and INFN, University of Rome “Tor Vergata,” Italy*

²*LTCL, Télécom Paris, IP Paris, France*

³*Inria and École Polytechnique, IP Paris, France*



(Received 21 November 2024; accepted 26 March 2025; published 10 April 2025)

This work studies an *a posteriori* data-driven approach (known as *solver-in-the-loop*) for subgrid modeling of a shell model for turbulence. This approach takes advantage of the *differentiable physics* paradigm of deep learning, allowing a neural network model to interact with the differential equation solver over time during the training process. The closure model is, then, naturally exposed to *equations-informed* input distributions by accounting for prior corrections over the temporal evolution in training. Such a characteristic makes this approach depart from the conventional *a priori* instantaneous training paradigm and often leads to a more accurate and stable closure model. Our study demonstrates that the closure learned via this *a posteriori* approach is able to reproduce high-order statistical moments of interest also in closures of high Reynolds number turbulence. Moreover, we investigate the performance of the learned model by experimenting with the effect of unrolling in time, which has remained for the most part unexplored in the literature. Finally, we discuss potential extensions of this approach to Navier-Stokes equations.

DOI: [10.1103/PhysRevFluids.10.044602](https://doi.org/10.1103/PhysRevFluids.10.044602)

I. INTRODUCTION

Three-dimensional turbulence is a complex, multiscale phenomenon that arises when the nonlinear transport terms in the Navier-Stokes (NS) equations dominate over viscous damping. The behavior of turbulent flows is governed by the Reynolds number, $Re = u_0 l_0 / \nu$, where u_0 represents the characteristic velocity, l_0 the typical length scale, and ν the kinematic viscosity. At high Re , turbulence exhibits a range of nontrivial behaviors, including non-Gaussian statistics and intermittent dynamics [1]. In three-dimensional (3D) turbulence, there is a nonlinear energy cascade, from large to small scales, where it is eventually dissipated through viscous friction [2].

Accurately resolving 3D turbulence is extremely computationally expensive. The degrees of freedom (DOF) scale as a power law of the Reynolds number, $\#_{DOF} \propto Re^{9/4}$ so studying extremely high Reynolds numbers numerically through primitive Navier-Stokes (NS) equations is often not possible. Modeling is needed. A central challenge in turbulence modeling, which has attracted much interest from both theoretical and applied researchers, is large eddy simulation (LES) subgrid scale (SGS) modeling [3–6]. LES reduces the degrees of freedom encountered in a fully resolved simulation by placing a filter at a certain wave number, k_c , and only resolving for $k < k_c$. This means that the so-called subgrid scales, $k > k_c$, need to be modeled. In contrast to what happens in other PDEs set-ups for fluids, such as, e.g., one-dimensional (1D) Kuramoto-Sivashinsky equations, 1D Burgers equations, and two-dimensional (2D) Navier-Stokes equations, modeling turbulence in 3D is theoretically more challenging because of the strong chaotic, out-of-equilibrium, and

*Contact author: andre.freitas@roma2.infn.it

non-Gaussian nature of high wave numbers, subgrid statistics, resulting in multifractal energy dissipation, and extreme subgrid energy transfer fluctuations [1]. Furthermore, from a more theoretical and fundamental point of view, the presence of anomalous scaling laws implies a breaking of self-similarity and the existence of a nontrivial dependency of the subgrid model from k_c [7,8]. In many applied cases, the cutoff wave number cannot be fixed and must be varied (*increased*) to improve fidelity of the resolved scale behavior. As a result, a comprehensive theoretical framework defining the statistical properties of the subgrid scale model in 3D turbulence is still missing.

In this paper, we focus on one specific theoretical aspect of the LES approach, connected to the subgrid scale anomalous statistical behavior. To do that, we need to study the effects of modeling when the k_c falls well inside the inertial range, and the nonlinear energy transfer is strongly non-Gaussian. In contrast to the more established phenomenology-based models [3], we will use a machine learning closure, inspired by the complexity of the modeling task and by recent promising results [9–14]. The main goal is to attack with high accuracy questions connected to the fidelity of the model to reproduce extreme SGS energy transfer events. No model is perfect, and one expects that extreme rare events are more sensitive to biases. The need to have high k_c (to observe non-Gaussian fluctuations) and very large statistics (for the data-driven approach) makes this study impossible in 3D turbulence, where most of the machine learning LES are limited to very small resolution (up to 128^3 or 256^3) and, consequently, by a very small departure from quasi Gaussian statistics. The only alternative framework where to study these questions is using shell models of turbulence, where only a few degrees of freedom are preserved for a set of logarithmically equispaced wave numbers, $k_n = k_0 \lambda^n$, where $\lambda = 2$ usually [15]. Models such as the Sabra model [16] have successfully replicated key statistical properties of turbulence, including intermittency, strongly non-Gaussian fluctuations, and anomalous scaling exponents. Shell models have been successfully used to study statistical properties of many turbulent fluid configurations, including rotating turbulence [17], thermal convection [18], superfluids [19], MHD turbulence [20], helical turbulence [21], and passive scalars [22], to cite just a few. Shell models have also been used to study fundamental properties of NS equations, connected to spontaneous stochasticity [23], effects of thermal noise [24], existence of solutions [25], instantons [26], and many more.

In this paper, we develop a deep learning-based SGS closure for shell models of turbulence. Our approach employs an *a posteriori* training technique known as *solver-in-the-loop*. This method incorporates a differentiable solver for the governing equations of a physical system directly into the learning process of a deep neural network tasked with learning the closure. We demonstrate that this approach yields closures that are more stable and perform better than those trained using the traditional static *a priori* and *instantaneous* paradigm. Additionally, we investigate the concept of the ideal *time in the loop*, a critical aspect that is often overlooked in the literature employing this methodology, and attempt to relate it to a relevant physical quantity.

In Sec. II, we review prior research on subgrid-scale modeling in LES, emphasizing machine learning closures. We pay particular attention to studies involving shell models and those utilizing differentiable solvers or unrolled training. In Sec. III, we discuss the closure of turbulence shell models within the LES framework and describe our solver-in-the-loop approach to closure in detail. In Sec. IV, we present and discuss the outputs of our trained models. Finally, Sec. V summarizes our findings and outlines potential future research directions.

II. RELATED WORK

Machine learning, and deep learning in particular, has seen wide adoption in fluid dynamics, as highlighted in several comprehensive reviews [27]. Generally, machine learning is applied in fluid dynamics either to fully replace a complex system with a surrogate model or to augment existing models by addressing unresolved scales or processes. LES closure falls into the latter category and has drawn significant interest from researchers. Recent studies have explored various approaches, including deep learning [9–12,28] as well as multiagent and deep reinforcement learning [13,14]. For a detailed perspective on data-driven turbulence closure, readers are referred to the review by

Duraisamy [29]. State-of-the-art ML tools are not yet able to tackle LES models for highly turbulent flows in the regime where the cutoff wave number is high enough to see the strong departure from Gaussianity. This is because of a combination of lack of computational power and/or accuracy, and lack of training data. These questions can be addressed in a quantitative way only in shell models, as of now. One of the first contributions of LES closure in the context of shell models of turbulence comes from Biferale *et al.* [30], who developed a theoretical framework to define an optimal subgrid closure. This phenomenological-based closure stands as a good comparison basis for new approaches. More recently, there has been a noticeable shift toward data-driven techniques. Ortali *et al.* [31] made important progress by using a deep recurrent neural network integrated within the time integrator scheme to close the system. Their approach yielded excellent results, especially in capturing both Eulerian and Lagrangian statistics. Another interesting approach is by Domingues Lemos *et al.* [32], who used a probabilistic method, specifically a mixture of Gaussians, to close the system. This added a new layer of complexity to LES closure strategies by taking into account the inherent probabilistic nature of the closure.

Among these approaches, Ortali *et al.*'s method is particularly interesting for us because it achieved the best results and it is the only one based on deep learning. Since they used an architecture with a memory component, they were able to effectively capture the time history effects in the closure. However, they used an *a priori* training approach and, as such, they did not fully account for the compounding effects of model errors over time. Addressing this issue would require unrolling the training process over time.

The concept of unrolling training in time with differentiable solvers was introduced by Um *et al.* in 2020 [33], under the term *solver-in-the-loop*, particularly for correcting errors of numerical solvers. This innovative approach allows for a nearest neighbor (NN) to interact with a differential equation solver for many time steps before performing backpropagation, exposing the NN to (more) correct input distributions, therefore improving the performance of the model when faced with the common distribution/data shift seen in the deployment of these kind of autoregressive models. A key advantage of this method is its reliance on automatic differentiation (AD) frameworks when developing the solver, which allow the gradients to also flow through the solver during backpropagation, leading to more precise unrolled gradients. Writing physical solvers using the AD framework is what is now commonly referred to as *differentiable physics*. More recently, List *et al.* [34] studied extensively the benefits of unrolling in time during training compared to a static instantaneous approach (*a priori* training), as well as the benefits of differentiability in the solver.

Another way to look at the benefits of different training schemes as well as different architectures is through the lens of inductive biases. In machine learning applied to science, models span a spectrum from those that rely almost entirely on data to those heavily informed by physical principles. At one end, fully connected networks with a non-physics-based loss function exemplify purely data-driven approaches, learning patterns directly from data without any built-in assumptions about the underlying system. Moving along the spectrum, convolutional networks [35] add some inductive biases, such as the assumption of locality and translation invariance, which are particularly effective in image processing. Further along, equivariant networks incorporate symmetries specific to the problem, like rotational symmetry, making them more specialized and efficient. Neural ordinary differential equations push this further by integrating differential equations into the model, embedding a continuous-time understanding of dynamics. Finally, at the most inductive end, models based on *solver-in-the-loop* approach or physics informed NNs, are tightly constrained by well-established physical laws. These models not only learn from data but also ensure that their predictions adhere to known physical principles, making them particularly valuable for complex scientific problems where adherence to physical laws is important. Moreover, they are able to learn with less data than purely data-driven models and tend to generalize better.

Other researchers have explored the use of differentiable solvers in combination with DL for LES closure. Notably, Sirignano *et al.* [36] applied this approach to 3D homogeneous isotropic turbulence (HIT) (at resolution 64^3), while Shankar *et al.* [37,38] utilized it for the Burgers equation (at resolutions 64–512) and 2D HIT (at resolution 64^2). These efforts do not, alas, extend to very high

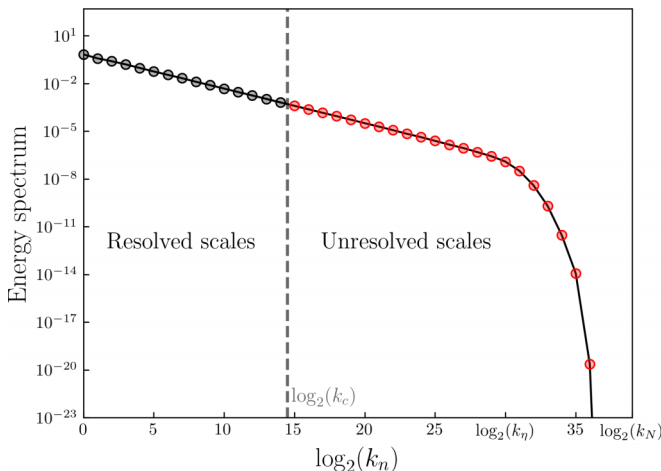


FIG. 1. Energy spectrum showing the large eddy simulation modeling problem. A cutoff is placed in the inertial range, in our case $N_c = 14$. The scales prior to the cutoff are resolved, whereas the one after are unresolved. The influence of the unresolved scales on the resolved ones needs to be modeled. In the case of the Sabra shell model of turbulence, which only has nearest and second-nearest-neighbor interactions, this means the shells $N_c + 1$ and $N_c + 2$ require modeling.

Reynolds numbers nor address the intense and multifractal non-Gaussian statistics typical of real turbulence (2D NSE in the forward enstrophy regime are even globally smooth). High Reynolds number turbulence presents unique challenges, and it is in this context that shell models become particularly valuable, offering a more tractable framework to study this phenomenon. This is where we believe a research gap exists, and our work aims to address this gap.

III. SHELL MODELS CLOSURE

Shell models mimic the dynamics of the energy cascade in three-dimensional homogeneous isotropic turbulence via a system of coupled nonlinear complex-valued ODEs describing the evolution of the velocity field on a set of wave number k_n logarithmically equispaced. In this work, we consider the Sabra model [16], for which the governing equations are

$$\frac{du_n}{dt} = i(ak_{n+1}u_{n+2}u_{n+1}^* + bk_nu_{n+1}u_{n-1}^* - ck_{n-1}u_{n-1}u_{n-2}) - \nu k_n^2 u_n + f_n, \quad (1)$$

where $n = 0, \dots, N$, $k_n = 2^n$, and $u_n \in \mathbb{C}$. Looking at the right-hand side we can see that, similar to the NSE in Fourier space, we have a nonlinear convective term (which similar to NSE defines the coupling among wave numbers; in the shell models only two-away neighboring interactions are considered) which is the trigger of the energy cascade mechanism, a quadratic dissipative term that dissipates energy at small scales and a forcing term which injects energy at the larger scales.

In a fully resolved system, the number of shells N is determined by the physics of the system. For a higher Reynolds number, the dissipative Kolmogorov length scale, k_η , will be at a large wave number and as such we have to consider enough shell to resolve it, $k_N > k_\eta$, see Fig. 1. The LES formulation in shells models is similar to a Galerkin Fourier truncation where we consider shells only up until the cutoff wave number k_{N_c} defined by the cutoff shell N_c where $N_c \ll N$ and it is usually somewhere in the inertial range. To close this reduced model, we need to provide a model for the two shells right after the cutoff u_{N_c+1} and u_{N_c+2} . This is depicted below and can be visualized

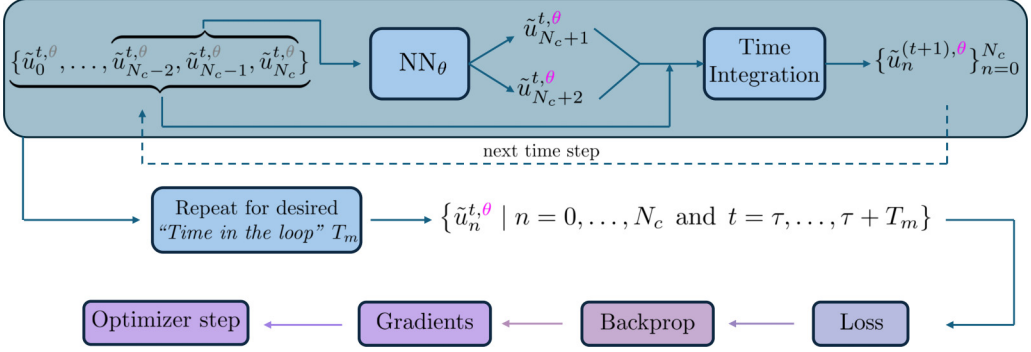


FIG. 2. Schematic representation of the LES-NN closure, illustrating how the neural network provides the necessary shells to close the system. Starting at time t , the NN takes as input the last three shells before the cutoff (this locally fixes the flux) $\{\tilde{u}_{N_c-2}^{t,\theta}, \tilde{u}_{N_c-1}^{t,\theta}, \tilde{u}_{N_c}^{t,\theta}\}$ and outputs the two shells after $\{\tilde{u}_{N_c+1}^{t,\theta}, \tilde{u}_{N_c+2}^{t,\theta}\}$ (in the image θ in gray denotes an implicit relation with the NN, whereas in magenta denotes an explicit one). This is enough to close the governing equations and evolve them in time to obtain the new state space at time. This process is repeated for a desired *time in the loop*. The resulting velocity field will be used to compute the loss (mean-squared error between the prediction and the ground truth). Backpropagation is applied to compute gradients, followed by an optimization step to update the NN.

in Fig. 1. We denote the fully resolved model as u , while the LES model is represented by $\tilde{u}$.

$$\begin{aligned} \text{Fully resolved model} & \begin{cases} u_{-1} = u_{-2} = 0, \\ u_{N+1} = u_{N+2} = 0, \end{cases} \\ \text{Large eddy simulation} & \begin{cases} \tilde{u}_{-1} = \tilde{u}_{-2} = 0, \\ \tilde{u}_{N_c+1} = \text{unknown, requires modeling}, \\ \tilde{u}_{N_c+2} = \text{unknown, requires modeling}. \end{cases} \end{aligned}$$

Now, we will introduce our LES-NN model as well as the basis of comparison, the ground truth (GT). The GT is simply the integration of the fully resolved system to generate training and testing data. This system is integrated over a long period to ensure that sufficient data is available to accurately compute the high-order moments of interest. Both the GT and LES-NN are integrated in time using a fourth-order Runge-Kutta (RK4) scheme with the viscous term integrated explicitly. However, different time steps are of course used: the GT is integrated with one much smaller than the LES-NN model to ensure the Kolmogorov scale (N_η) is resolved.

While both systems have the same time integration method, the shell models are different. The GT resolves the $\{u_0, \dots, u_N\}$ using the governing equations. The LES-NN uses the reduced solver that resolved $\{\tilde{u}_0, \dots, \tilde{u}_{N_c}\}$ using the governing equations and then a neural network at each time step estimates $\tilde{u}_{N_c+1}$ and $\tilde{u}_{N_c+2}$, therefore closing the system. This is shown in Fig. 2. As input to the neural network, we provide the three shells preceding the cutoff, which is sufficient to close the flux locally. Using fewer shells results in significantly poorer performance, while including more shells offers no noticeable improvement.

One implementation is purposely agnostic to the time integrator used: we integrate the missing terms from the governing equations for $\tilde{u}_{N_c-1}$ and $\tilde{u}_{N_c-2}$ explicitly as shown in Eqs. (2) and (3).

$$\frac{d\tilde{u}_{N_c-1}^\theta}{dt} = i \left(\underbrace{ak_{N_c}\tilde{u}_{N_c+1}^\theta\tilde{u}_{N_c}^*}_{\text{Integrated explicitly}} + \underbrace{bk_{N_c-1}\tilde{u}_{N_c}\tilde{u}_{N_c-2}^* - ck_{N_c-2}\tilde{u}_{N_c-2}\tilde{u}_{N_c-3}}_{\text{Integrated with RK4}} \right) - \nu k_{N_c-1}^2 \tilde{u}_{N_c-1}, \quad (2)$$

$$\frac{d\tilde{u}_{N_c}^\theta}{dt} = i \left(\underbrace{ak_{N_c+1}\tilde{u}_{N_c+2}^\theta \tilde{u}_{N_c+1}^{*,\theta} + bk_{N_c}\tilde{u}_{N_c+1}^\theta \tilde{u}_{N_c-1}^*}_{\text{Integrated explicitly}} - \underbrace{ck_{N_c-1}\tilde{u}_{N_c-1}\tilde{u}_{N_c-2}}_{\text{Integrated with RK4}} \right) - vk_{N_c}^2 \tilde{u}_{N_c}. \quad (3)$$

Algorithm 1 shows the training loop function. For ease of understanding, the shells between the cutoff are shown as $\tilde{u}^<$ and the ones after as $\tilde{u}^>$. As it can be seen, the gradients are also being propagated through the solver operations (RK4, which calls the rest of the solver functions). It is also possible to perform unrolled training in the case where the solver is not differentiable, but then one needs to either stop the gradient flow during backpropagation whenever the solver is called (which in the end will lead to worse quality gradients) or to provide by hand the AD primitives. By having a differentiable solver, we are able to bypass these two disadvantages and leave all of the hard work to the AD framework—the obvious downside of this approach is having to (re)write the solver in an AD framework, which also comes with a few caveats compared to regular non-AD framework programming. In one training iteration, we evolve the system for `msteps`, which is a hyperparameter. This represents the time that we evolve the system before backpropagating the gradients, i.e., before updating the NN weights.

The architecture used for our neural network is the multilayer perceptron (MLP) [39] with rectified linear unit (ReLU) as the activation function. The number of trainable parameters used in the MLP varied during our studies between 1×10^5 and 4×10^5 , with the latter used for the results presented here. The loss used is the mean-squared error (MSE) between the prediction of the reduced system LES-NN, $\tilde{u}$, and the ground truth, u , i.e.,

$$\mathcal{L} = \frac{1}{N_{\text{Loss}}} \sum_{n=1}^{N_{\text{Loss}}} \frac{\|u - \tilde{u}\|_{bs, T_m}^2}{\sqrt{\|u\|_{bs, T_m}^2} \sqrt{\|\tilde{u}\|_{bs, T_m}^2}}, \quad (4)$$

where $\|u\|_{bs, T_m}^2 = \sum_{b=1}^{bs} \sum_{t=\tau_b}^{\tau_b+T_m} |u_{n,t,b}|^2$, T_m denotes the time in the loop, bs the batch size and N_{loss} is the number of shells considered in the loss function, which in our case is equal to six and these are the shells before the cutoff.

Table I shows the parameters used in the numerical experiments shown in the following section. Regarding the forcing, the first two shells are forced constantly in time with the magnitudes $f_0 = \epsilon$ and $f_1 = 0.7\epsilon$. This forcing ensures zero helicity flux [16].

ALGORITHM 1. Training Loop Algorithm (a single training iteration).

```

1: Initialize Gradient Tape
2:  $\tilde{u} \leftarrow \tilde{u}_0$   $\triangleright$  batch of ICs selected randomly from dataset
3: for  $t = 0$  to  $msteps - 1$  do
4:    $\tilde{u}_t^{>,\theta} \leftarrow \text{NN}_\theta(\tilde{u}_t^{<})$ 
5:    $\tilde{u}_{t+1}^{<} \leftarrow \text{RK4}(\tilde{u}_t^{<})$ 
6:    $\mathcal{C}_{N_c-1} \leftarrow \Delta \tilde{i}(ak_{N_c}\tilde{u}_{N_c+1}^\theta \tilde{u}_{N_c}^*)$ 
7:    $\mathcal{C}_{N_c} \leftarrow \Delta \tilde{i}(ak_{N_c+1}\tilde{u}_{N_c+2}^\theta \tilde{u}_{N_c+1}^{*,\theta} + bk_{N_c}\tilde{u}_{N_c+1}^\theta \tilde{u}_{N_c-1}^*)$ 
8:    $\mathcal{C} \leftarrow \text{concatenate}(\mathcal{C}_{N_c+1}, \mathcal{C}_{N_c+2})$ 
9:    $\tilde{u}_{t+1}^{<} \leftarrow \tilde{u}_{t+1}^{<} + \mathcal{C}$ 
10: end for
11: Compute Loss
12: Compute Gradients
13: Apply Gradients
    
```

TABLE I. Values of the parameters of the numerical experiments.

Parameter	Value	Description
ν	1×10^{-12}	Viscosity
Re	$\approx 10^{12}$	Reynolds number
ϵ	0.5	Forcing
N	40	Number of shells
N_η	30	Kolmogorov scale
N_c	14	Subgrid cutoff scale
τ_0	7.553×10^{-1}	Eddy turnover time for the integral scale
τ_η	1.8367×10^{-6}	Eddy turnover time for the dissipative scale
Δt	1×10^{-8}	Time step of GT
$\Delta \tilde{t}$	1×10^{-5}	Time step of LES-NN model
N_{data}	256	Number of initial conditions of dataset
N_{batch}	1024	Batch size for training
T_{train}	$1.65 \tau_0$	Integration time of training dataset
T_{test}	$3.31 \tau_0$	Integration time of test dataset

IV. RESULTS

In the following, we present the results from our model and how they compare to the ground truth. In some of the results, we also compare them with state-of-the-art DL closures as well as phenomenological ones.

Figure 3 shows the flatness of different orders, from $F^{(4)}$ and $F^{(10)}$, with respect to the shell index. The flatness is computed in terms of the Eulerian structure functions as

$$F_n^{(p)} = \frac{S_n^{(p)}}{(S_n^{(2)})^{\frac{p}{2}}}, \quad (5)$$

where the Eulerian structure functions are expressed:

$$S_n^{(p)} = \langle |u_n|^p \rangle_t, \quad (6)$$

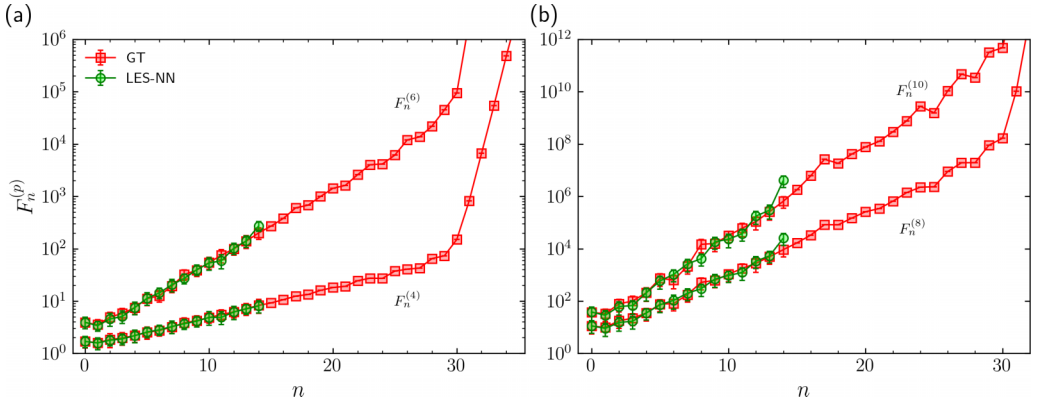


FIG. 3. Flatness of different orders computed in terms of the Eulerian structure functions by Eq. (5): (a) flatness of order 4 and 6; (b) flatness of order 8 and 10. Error bars computed by dividing the dataset into chunks, computing the individual chunk's statistics and from here estimate the standard deviation. The error bars are only shown until the cutoff scale.

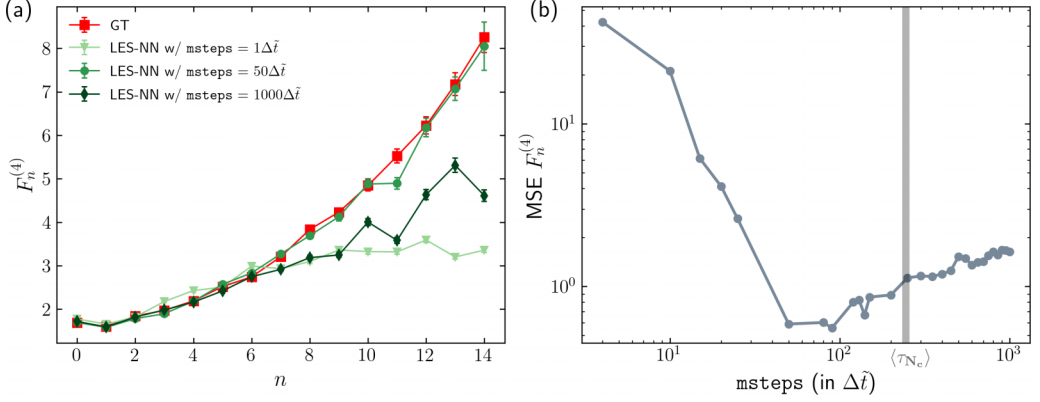


FIG. 4. (a) Fourth-order flatness for different time in the loop ($msteps$) and compared with the ground truth (error bars represent the standard deviation). (b) Mean-squared error of the fourth-order flatness [Eq. (7)] for different time in the loop.

with $\langle \cdot \rangle$ representing the averaging operator. The lower-order flatnesses show a good agreement with the ground truth. As the order increases, we start to notice some deviations, especially near the cutoff. Despite these deviations, the results remain promising, as these higher-order moments are nontrivial to reproduce correctly, and phenomenological closures fail to capture them accurately.

Figure 4 attempts to determine the optimal time in the loop. On the left, $F_n^{(4)}$ is shown for different times in the loop (the $msteps$ variable used in Algorithm 1) $1\Delta t$, $50\Delta t$, and $1000\Delta t$, where one time step in the loop corresponds to the *a priori* training paradigm. We can see that the best results are obtained with a value of $msteps = 50\Delta t$, while both $msteps = 1\Delta t$ and $msteps = 1000\Delta t$ perform poorly in comparison. In the subfigure on the right, we show a continuation of this analysis, where we plot the MSE of $F_n^{(4)}$, given by

$$\text{MSE}(F_n^{(4)}) = \frac{\sum_{n=0}^{N_c} |F_{n_{GT}}^{(4)} - F_{n_{LES}}^{(4)}|^2}{N_c}, \quad (7)$$

with respect to the time in the loop. This allows us to better understand the effect of the time in the loop in the effectiveness of the training procedure and how it impacts the final performance of the learned closure. There is a benefit in increasing the time in the loop until around $100\Delta t$ in the loop. Keeping on increasing after this threshold increases the error. The highest MSE occurs with instantaneous evaluation, i.e., when $msteps = 1$.

We saw that there is a clear effect from the duration in the loop during training in the performance of the model. *A priori*, we expect that the optimal loop time will be a fraction of the eddy turnover time of the fastest shell included in the loss. Since we use a loss function that measures the difference between velocity fields, exceeding the eddy turnover time of the fastest shell with a high $msteps$ value causes the signals (GT and our model) to decorrelate, making the loss less meaningful. Our focus is on achieving a statistically accurate closure rather than synchronizing with the GT, which is unrealistic. Therefore, the ideal loop time is expected to be a fraction of τ_{N_c} . Exceeding this value smooths out the dynamics of the fastest shells, pushing the model to track the moving average of the GT rather than its exact behavior.

To better understand this relation between time scales of the system and $msteps$, we show the probability density function (pdf) of the eddy turnover time τ_n for the shells considered in the loss in Fig. 5, computed as

$$\tau_n = \frac{1}{k_n \sqrt{\langle |u_n|^2 \rangle_t}}, \quad (8)$$

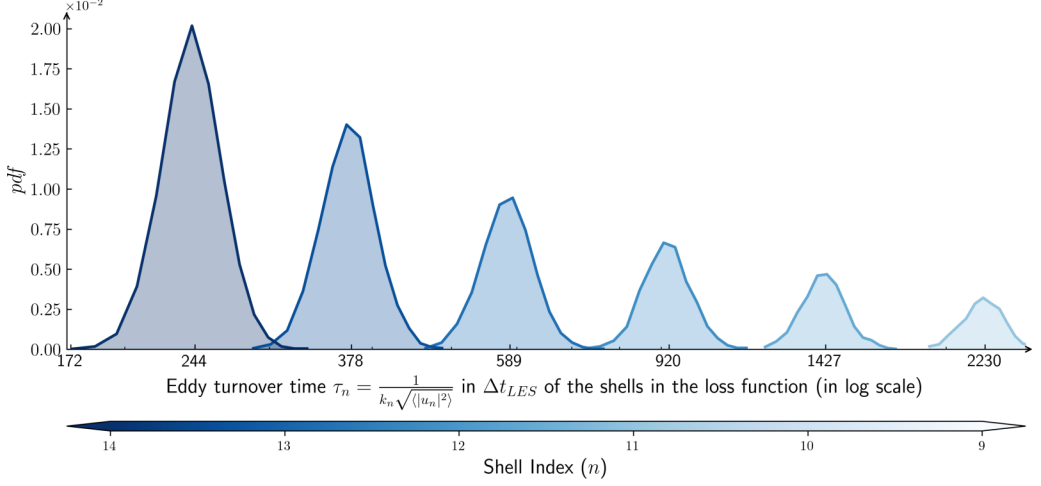


FIG. 5. pdf of the eddy turnover time of the shells used in the loss function.

where the pdf is obtained considering a time signal of u_n for various initial conditions.

Looking back at Fig. 4, when we examine the MSE of $F_n^{(4)}$, we see that the optimal m steps value corresponds to a fraction of the eddy turnover time of the cutoff shell, $\langle \tau_{N_c} \rangle = 244$, with the ideal value being around m steps = 100, or approximately $0.41 \langle \tau_{N_c} \rangle$. This analysis shows the benefit of using the *solver-in-the-loop* approach versus the conventional static paradigm and helps understand the physicality of the optimal time in the loop. Throughout the rest of the paper, we will try to keep making similar analyses as we did here for the flatness, for other quantities, as to validate our hypothesis.

Figure 6 shows the Eulerian structure functions. The results from our closure align closely with the GT within error bars, though more noticeable deviations appear as the order increases and near

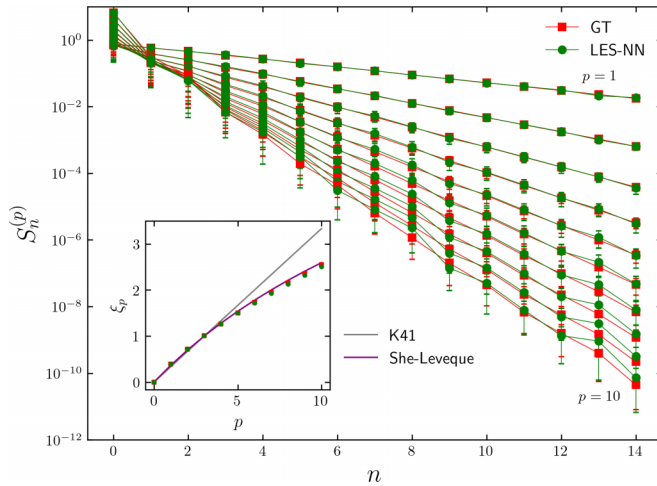


FIG. 6. Eulerian structure functions $S_n^{(p)} = \langle |u_n|^p \rangle_t$ vs shell index n , in lin-log scale, for orders p from 1 to 10 and with $N_c = 14$, comparison between ground truth (GT) and prediction (Pred). Inset plot: Anomalous scaling exponents ξ_p of the Eulerian structure functions $S_n^{(p)} \propto k_n^{-\xi_p}$ for the fully resolved model (GT), our model (LES-NN), the prediction from K41 theory [40] and the prediction from She-Leveque model [41].

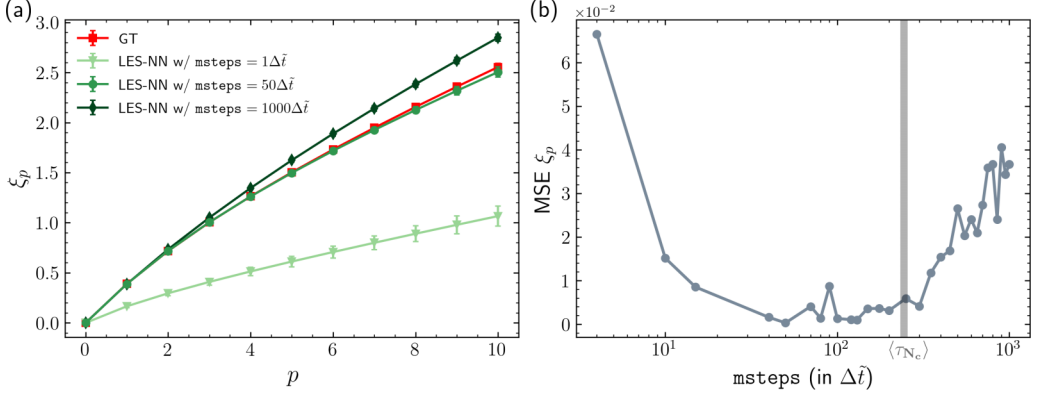


FIG. 7. (a) Anomalous scaling exponents ξ_p for different time in the loop ($msteps$) and compared with the ground truth. (b) MSE of ξ_p [Eq. (10)] for different time in the loop.

the cutoff. The error bars are estimated by splitting the datasets in chunks. We compute individual statistics for each chunk, report the average as the central point, and use the difference between the minimum and maximum as the error bar. To further verify our implementation, we show as an inset plot the anomalous scaling exponents ξ_p of the Eulerian structure functions:

$$S_n^{(p)} \propto k_n^{-\xi_p}. \quad (9)$$

Also here, we see an agreement with the GT similar to what we saw with the flatness.

Looking deeper into the anomalous scaling exponents, Fig. 7 shows on the left the comparison of this quantity for different time in the loop and on the right the MSE computed via Eq. (10). On the left-hand side, we see similar results as we saw before, where a value of $msteps = 50\Delta t$ performs best:

$$MSE(\xi_p) = \frac{\sum_{p=1}^{P=10} |\xi_{p_{GT}} - \xi_{p_{LES}}|^2}{P} \quad (10)$$

The MSE of the anomalous scaling exponents is even more expressive than the one of the flatness, as it incorporates statistical moments from $p = 1$ to $p = 10$ (and since it is not normalized, it gives more weight to higher order ones). Similar to the flatness case, the ideal loop time is a fraction of the eddy turnover time of the fastest shell. Deviating too much from this value, either higher or lower, results in an increase in MSE.

Shifting the perspective from Eulerian to Lagrangian, we now examine the Lagrangian structure functions, assessing whether our model accurately reproduces time correlations across various time lags. This is illustrated in Fig. 8.

The Lagrangian structure functions are computed as

$$L_\tau^{(p)} = \langle |u(t + \tau) - u(t)|^p \rangle_t, \quad (11)$$

where the Lagrangian signals are obtained by summing the real parts of all the shells $u(t) = \Re(\sum_n u_n(y))$. Analyzing the results, one can see that the model closely follows the scaling of the ground truth, even for small time lags and higher-order moments (within error bars), which are the most challenging to capture accurately.

In Fig. 9, we show another statistical quantity: the pdf of the real part of the velocity signals for different shells $n = 4, 9, 14$ normalized by the standard deviation, for both the model and the ground truth. We see that our closure has the correct effect on the resolved scales as we are able to correctly reproduce the Gaussian statistics of the large scales and more importantly the non-Gaussian statistics of the small scales, characterized by intermittency.

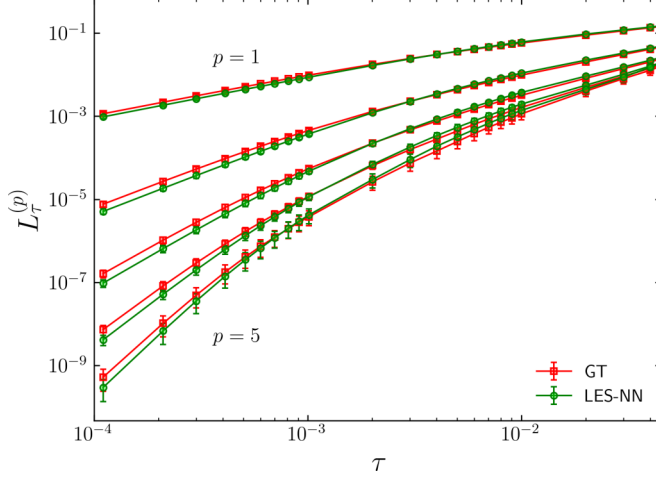


FIG. 8. Lagrangian structure functions of orders $p = 1, \dots, 5$ in log-log scale with the time lag, τ , on the x axis.

We aim also to compare our closure with other state-of-the-art closures. As such, in Fig. 10(a), we show the local slopes for the second-order Eulerian structure function, expressed through

$$\zeta_n^{(p)} = \frac{\log[S_{n+1}^{(p)}] - \log[S_n^{(p)}]}{\log[\lambda]}, \quad (12)$$

with respect to the shell index, for our model, the GT, the long short-term memory (LSTM) [42] approach from Ortali *et al.* [31] and a phenomenological closure from Biferale *et al.* [30]. The LSTM approach performs well overall although its accuracy decreases near the cutoff. Its memory component compensates for the static training, contributing to its robust performance. The phenomenological closure, *smk*, performs adequately in the mid-range but shows significant degradation near the boundaries. In contrast, our approach oscillates around the GT and achieves the best performance of the three models, particularly near the cutoff—where correctly reproducing the slopes is most challenging due to the more pronounced effect of the closure.

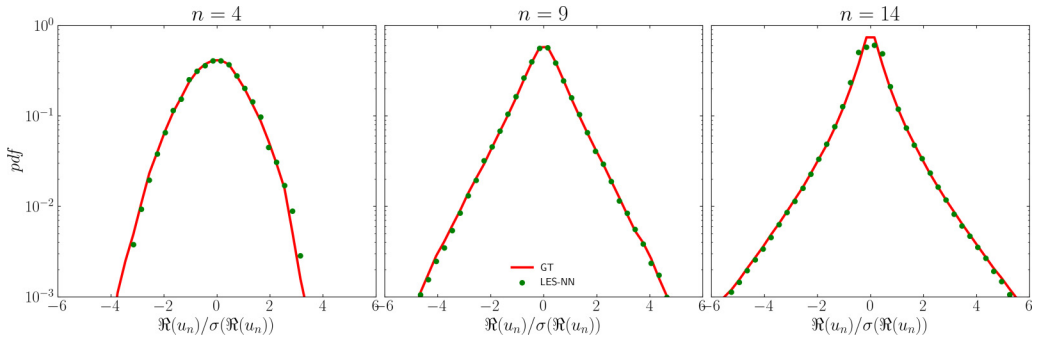


FIG. 9. pdf of the real part of the shells 4, 9, and 14 (cutoff shell), in log scale, normalized with the standard deviation $\Re(u_n)/\sigma(\Re(u_n))$ for the GT and prediction.

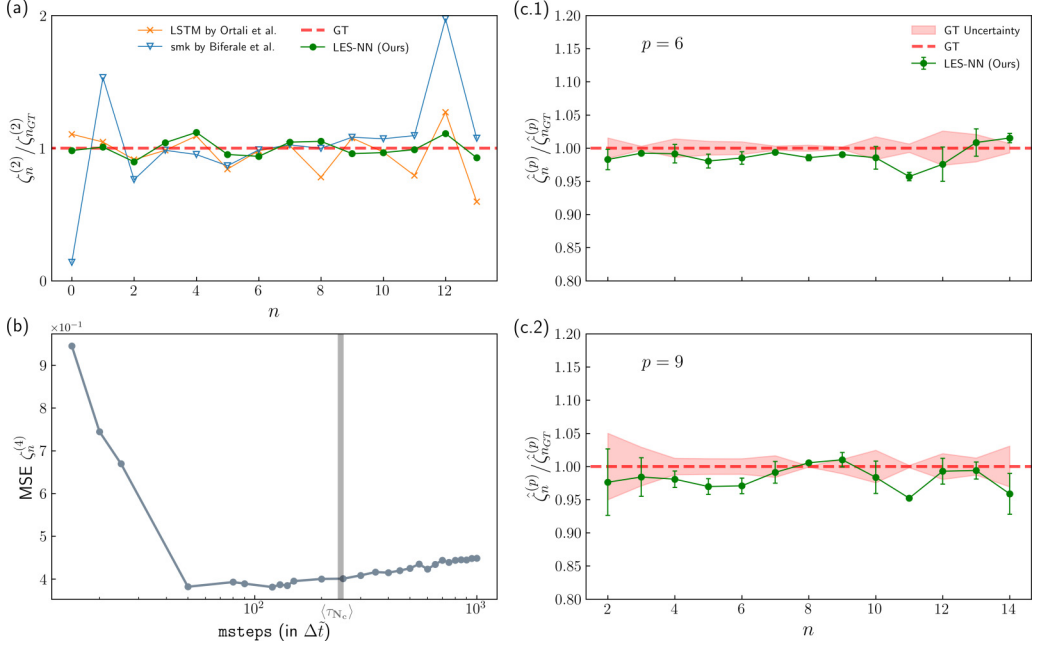


FIG. 10. (a) The local slopes (normalized by the ones of the GT) for the second order Eulerian structure functions [Eq. (12)] vs shell index n . Comparison between ground truth (GT), our model (LES-NN), the state-of-the-art LSTM approach by Ortali *et al.* [31] and a non-ML approach of an optimal subgrid closure scheme by Biferale *et al.* [30]. (b) MSE of $\zeta_p^{(4)}$ [Eq. (13)] for different time in the loop. (c) Normalized local slopes of the Eulerian structure functions computed using the triads [Eq. (14)] for $p = 6$ and $p = 9$. Remark: this expression to compute the Eulerian structure functions using the triads, unlike $\langle |u_n|^p \rangle$, is not prone to period-3 oscillations and as such the local slopes oscillate much less. This is why even for such high-order moments slopes as the ones in panels (c.1) and (c.2) the range of values is much closer to the GT than in panel (a).

Figure 10(b) shows the MSE of the local slopes of the fourth-order Eulerian structure function:

$$\text{MSE}(\zeta_n^{(4)}) = \frac{\sum_{n=0}^{N_c} |\zeta_{nGT}^{(4)} - \zeta_{nLES}^{(4)}|^2}{N_c - 1}. \quad (13)$$

The trend is similar to the one seen before, although here the increase in error with high $msteps$ values is not so severe as before. The way to interpret this trend is that for high $msteps$, the slope of $S_n^{(4)}$ is correct, but the total energy content is off (see the vertical shift of the structure functions). The intershell relations are preserved, but the absolute energy is inaccurate. This also explains why the anomalous scaling exponents showed the most difference out of the three MSE errors for high $msteps$: it considers very-high-order moments and gives them a considerable weight.

Last, in Fig. 10(c), we show the normalized local slopes of the Eulerian structure functions computed with respect to the triads. These structure functions are computed from Eq. (14). Unlike the ones from Eq. (6), these are not prone to period-3 oscillations. The slopes are computed using the same expression as before, Eq. (12). As expected, performance degrades as the cut-off is approached. Despite this, for such a sensitive quantity as the local slopes of structure functions, our model remains relatively close to the ground truth, with errors of less than 5%:

$$\hat{S}_n^{(p)} = \langle |u_{n-2}^* u_{n-1}^* u_n|^{\frac{p}{3}} \rangle_t. \quad (14)$$

As demonstrated in previous figures, our model exhibits some error relative to the ground truth. It is important to determine whether this error arises solely from the model's inherent limitations

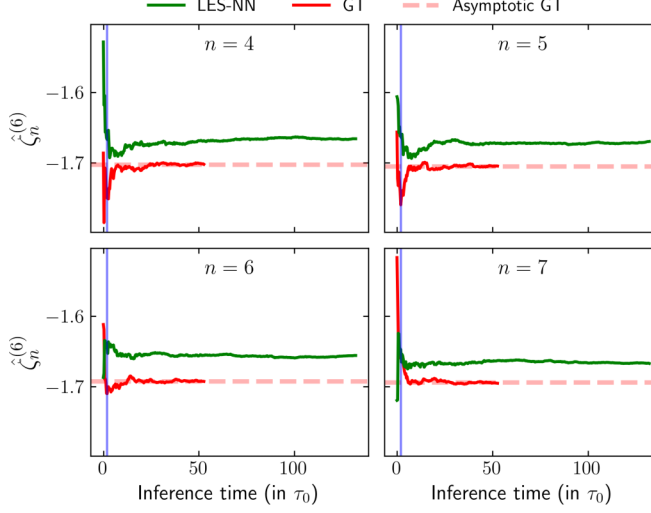


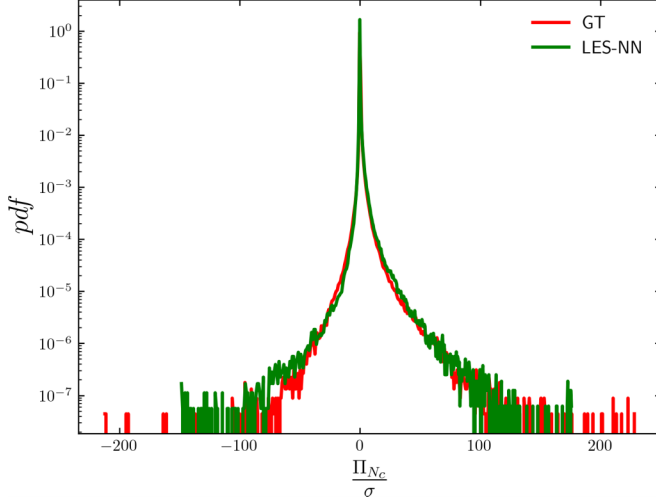
FIG. 11. Local slopes of the sixth-order Eulerian structure function computed as a function of the triads [Eq. (14)], $\hat{\zeta}_n^{(6)}$, with respect to the inference/deployment time (expressed in terms of τ_0). It is shown for $n = 4, 5, 6, 7$. Compared with the GT reference data. The vertical bar denotes the amount of data used for training data.

or if a significant statistical error is also present, potentially due to computing a given statistical observable from a limited sample size. Figure 11 explores this issue by showing how the local slope of the third-order Eulerian structure function computed using the triads [Eq. (14)] evolves over increasing deployment time, for shells $n = 4, 5, 6$, and 7 . The figure presents results for both our model (LES-NN) and the ground truth.

We observe that only a few eddy turnover times are needed to approach the asymptotic value, both for the ground truth and our model. This suggests that the errors highlighted throughout the paper are primarily due to intrinsic model limitations rather than statistical fluctuations (with the obvious caveat that error bars, when shown, refer to statistical errors). The vertical bar in the figure represents the amount of data used to train our model. Notably, the model remains stable even when deployed far beyond the time frame it was trained on. This stability naturally arises from our training methodology, where we explicitly constrain the time evolution based on the actual governing equations.

The GT is shown for less deployment time than the LES-NN model because it takes much longer to run. We benchmarked the time it takes to run them both on an NVIDIA A100 GPU, averaging over many realizations to ensure the validity of our results. We started with an ensemble of initial conditions and evolved them for one eddy turnover time of the slowest shell, τ_0 . On average, the GT took about 81 min, whereas the LES-NN only took 6 min.

This difference can be attributed to the higher time step used in the LES. Although the presence of the neural network introduces some computational overhead, our fully differentiable framework allows us to accelerate computations by utilizing graph mode and accelerated linear algebra (XLA) compilation [43]. Graph mode enables numerous optimizations at the compiler level, such as statically determining the values of tensors by combining constant nodes in the computation, commonly referred to as “constant folding.” XLA allows for the optimization of the computational graph. One such optimization is for example the separation of independent parts of a computation, enabling them to be processed across multiple threads or devices. This parallelism enhances performance significantly. Furthermore, XLA simplifies arithmetic operations by eliminating common subexpressions, leading to a more efficient execution of the model.


 FIG. 12. pdf of the flux at shell N_c [Eq. (15)].

To evaluate the correct reproduction of the energy fluxes given our closure, we show the pdf of the convective fluxes at the cutoff shell, Π_{N_c} in Fig. 12. Where Π_n is given by

$$\Pi_n = \Im[ak_{n+1}u_{n+2}u_{n+1}^*u_n^* + (b+a)k_nu_{n+1}u_n^*u_{n-1}^*]. \quad (15)$$

The results show a strong agreement with the fully resolved model. A positive value of this flux indicates a forward energy cascade at the cutoff shell, transferring energy to smaller scales. Conversely, a negative value is called backscatter, meaning energy flows from smaller scales to larger ones. This phenomenon is particularly challenging to model in subgrid-scale models, as improper handling of negative energy flux can lead to numerical instabilities. This is why phenomenological closures often avoid addressing backscatter. Similarly, some deep learning-based closures sidestep this issue to simplify training and ensure model stability.

Figure 13 shows a comparison of simulation results over a selected time interval. The top row depicts the large scales, where the dynamics between the ground truth and our model remain qualitatively similar until about $t = 0.5\tau_0$. After this point, the phases begin to decorrelate, especially for the smallest large-scale components (shown in darker colors). The small scales (bottom row) become fully decorrelated at around the same time, but it occurs more rapidly due to their shorter eddy turnover times.

It is unreasonable to expect a subgrid-scale model to maintain synchronization between the LES model and the GT for extended periods. The goal is simply to recover the statistical moments of the GT rather than precisely replicate its dynamic evolution.

The time step used for the LES, $\Delta\tilde{t} = 10^{-5}$, was chosen simply because it was the one used by Ortali *et al.* [31]. Given that our model performs best when trained for an unrolled time of approximately 250 LES time steps ($\approx \langle \tau_{N_c} \rangle$) or fewer, we were curious to see how the model would behave if we increased the time step up to the limit where only two steps are performed in the loop ($msteps = 2\Delta\tilde{t}$), with a time step of 10^{-3} . This analysis is presented in Fig. 14, where we plot the MSE of the flatness for various orders, ranging from 4 to 10, as a function of the LES time step.

Surprisingly, we observe that the model maintains very good performance even with a time step 10 times larger than what was used throughout the paper. It is important to note that a time step of $\Delta\tilde{t} = 10^{-4}$ is 10000 times larger than the ground-truth evolution's time step. This indicates that the neural network is not only learning the physical closure but also some numerical error associated with such coarse temporal dynamics. However, when the time step is further increased to $\Delta t = 5 \times 10^{-4}$ or $\Delta t = 10^{-3}$, the errors grow significantly, and a noticeable drop in performance

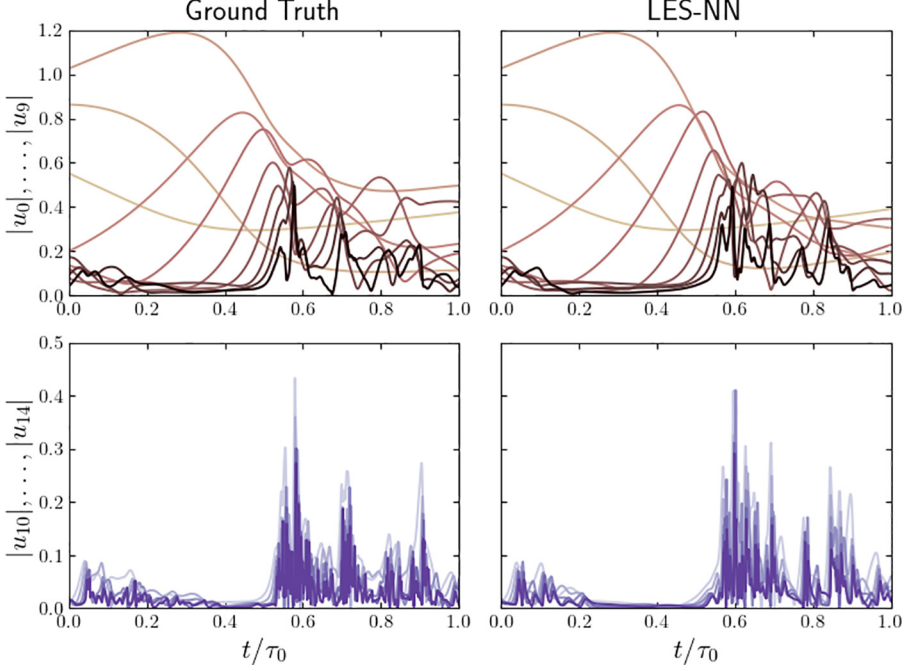


FIG. 13. Qualitative comparison between the GT and prediction in terms of the dynamics of the absolute value of the large scales, $|u_0|, \dots, |u_9|$ and the small scales, $|u_{10}|, \dots, |u_{14}|$.

occurs. This is due to two factors: the increasing influence of numerical errors, which makes the task more challenging for the neural network, and the reduced number of unrolled steps during training, as larger time steps reach the limit of τ_{N_c} more quickly.

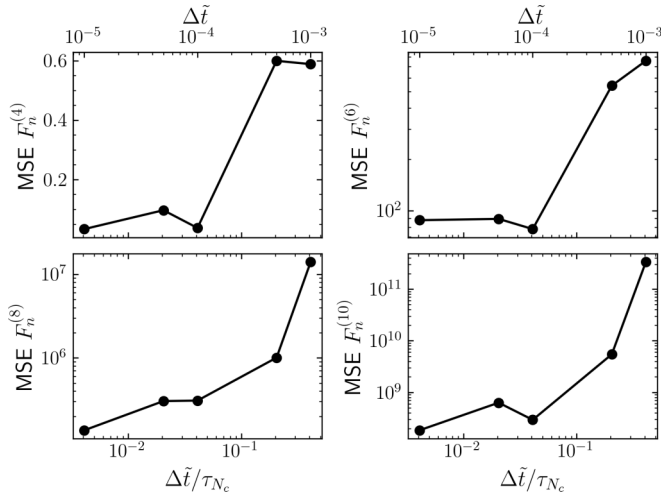


FIG. 14. Mean-squared error for the flatness of different orders, $F_n^{(4)}, \dots, F_n^{(10)}$, for LES-NN models trained with different time steps, $\Delta \tilde{t}$. The error is computed with respect to the ground truth. The time step is normalized by the time scale of the cutoff shell.

V. CONCLUSIONS

In this work, we have proposed a solver-in-the-loop approach to learning subgrid-scale closures in a shell model of turbulence. This methodology leverages the differentiable physics paradigm, allowing the neural network to interact with the solver during training and optimize the closure terms *a posteriori*. By incorporating unrolled solver interactions, we have demonstrated that our model outperforms traditional a-priori trained models in terms of stability and accuracy. Moreover, we show that our model is able to perform similarly or even outperform state-of-the-art deep learning approaches with complex architectures, despite relying on a simpler architecture.

Our results on shell models suggest that the optimal time-in-the-loop is closely tied to the eddy turnover time of the fastest shells included in the loss function. This time scale serves as an approximation of the Lyapunov time of the LES system. When using a loss function based on the difference between velocities, setting the time-in-the-loop beyond the Lyapunov time effectively attempts to synchronize two systems beyond their synchronization time. Moreover, as the time-in-the-loop increases, the gradients during backpropagation become more likely to either explode or vanish, a phenomenon related to the Lyapunov time, making gradient-based optimization increasingly unstable. We speculate that even if a different loss function was used, e.g., one based on difference in energy flux between the ground truth and the model, instead of difference in velocities, the optimal time-in-the-loop would not differ significantly from the one identified in our study. While such a loss function does not explicitly enforce trajectory synchronization and is therefore not directly constrained by the Lyapunov time, the gradients' quality deteriorates as the time-in-the-loop exceeds the Lyapunov time, which might lead to divergence in training or "worse" optimizer steps.

Extending these conclusions to the 3D NS introduces additional challenges, primarily due to the increased memory requirements. The general principles derived from our study may still hold, though this remains to be investigated.

Our study also provides relevant insights for future work using the solver-in-the-loop approach, particularly in how to *a priori* tune this time-in-the-loop hyperparameter. These insights are grounded in the physics of the system, allowing for better generalization across different physical models.

Beyond shell models, we believe this approach shows great potential for more complex systems, including Navier-Stokes equations, where unresolved scales play a much more complex role, making learning closure models more challenging. As such, future work includes extending this framework to NSE turbulence. A potential candidate is natural convection in 2D, where the presence of nontrivial multifractal scaling properties for temperature and Bolgiano scaling for velocity make the closure problem potentially as challenging as in 3D turbulence, still retaining a smaller degree of complexity. Additionally, the use of differentiable solvers opens up new possibilities for integrating physical priors more deeply into machine learning frameworks, potentially further improving generalization and data efficiency. This insight is not limited to subgrid-scale closure in the context of LES but can also benefit other problems in fluid dynamics and more generally in science where machine learning can provide solutions.

The solver-in-the-loop methodology can also be compared to reinforcement learning (RL) approaches, which similarly aim to optimize decision making through iterative feedback. While model-free RL typically involves exploring a vast action space and learning from trial and error, our approach directly integrates the physics of the problem, leveraging the differentiable nature of the solver to guide the neural network training. However, when considering model-based RL, the distinctions between our solver-in-the-loop approach and RL become less clear. Model-based RL uses an explicit model of the environment to predict future states and optimize actions, similar to how our methodology utilizes a differentiable physics model during training. The solver in the loop approach is conceptually similar to model-based deep RL, i.e. when the policy is parameterized by a (deep) neural network. This overlap raises interesting questions about the advantages and disadvantages of each approach.

In conclusion, the solver-in-the-loop approach presents a robust and flexible method for addressing subgrid-scale modeling challenges in turbulence using deep learning. We believe it provides a valuable perspective for combining machine learning with differentiable physics to tackle complex, multiscale systems.

ACKNOWLEDGMENTS

The authors benefited from discussions with M. Sbragaglia. This research was supported by European Union's HORIZON MSCA Doctoral Networks programme under Grant Agreement No. 101072344, project AQTIVATE (Advanced computing, Quantum algorithms and data-driven Approaches for science, Technology and Engineering), the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme Smart-TURB (Grant Agreement No. 882340), and through an Inria Chair.

-
- [1] U. Frisch, *Turbulence: The legacy of A. N. Kolmogorov* (Cambridge University Press, Cambridge, UK, 1995).
 - [2] A. Alexakis and L. Biferale, Cascades and transitions in turbulent flows, *Phys. Rep.* **767**, 1 (2018).
 - [3] C. Meneveau and J. Katz, Scale-invariance and turbulence models for large-eddy simulation, *Annu. Rev. Fluid Mech.* **32**, 1 (2000).
 - [4] P. C. M. Lesieur and O. Met  is, *Large-Eddy Simulations of Turbulence* (Cambridge University Press, Cambridge, UK, 2005).
 - [5] S. B. Pope, *Turbulent Flows* (IOP Publishing, Bristol, UK, 2001).
 - [6] P. Sagaut, *Large Eddy Simulation for Incompressible Flows: An Introduction* (Springer, Berlin, 2006).
 - [7] A. A. Mailybaev, Hidden scale invariance of intermittent turbulence in a shell model, *Phys. Rev. Fluids* **6**, L012601 (2021).
 - [8] A. A. Mailybaev and S. Thalabard, Hidden scale invariance in Navier–Stokes intermittency, *Philos. Trans. R. Soc. A* **380**, 20210098 (2022).
 - [9] R. Maulik, O. San, A. Rasheed, and P. Vedula, Sub-grid modelling for two-dimensional turbulence using neural networks, *J. Fluid Mech.* **858**, 122 (2019).
 - [10] R. Maulik, O. San, J. D. Jacob, and C. Crick, Sub-grid scale model classification and blending through deep learning, *J. Fluid Mech.*, **870**, 784 (2019).
 - [11] A. Beck, D. Flad, and C.-D. Munz, Deep neural networks for data-driven LES closure models, *J. Comput. Phys.* **398**, 108910 (2019).
 - [12] H. Frezat, G. Balarac, J. L. Sommer, R. Fablet, and R. Lguensat, Physical invariance in neural networks for subgrid-scale scalar flux modeling, *Phys. Rev. Fluids* **6**, 024607 (2021).
 - [13] G. Novati, H. L. de Laroussilhe, and P. Koumoutsakos, Automating turbulence modelling by multi-agent reinforcement learning, *Nat. Mach. Intell.* **3**, 87 (2021).
 - [14] M. Kurz, P. Offenhauser, and A. Beck, Deep reinforcement learning for turbulence modeling in large eddy simulations, *Int. J. Heat Fluid Flow* **99**, 109094 (2023).
 - [15] L. Biferale, Shell models of energy cascade in turbulence, *Annu. Rev. Fluid Mech.* **35**, 441 (2003).
 - [16] V. S. L'vov, E. Podivilov, A. Pomyalov, I. Procaccia, and D. Vandembroucq, Improved shell model of turbulence, *Phys. Rev. E* **58**, 1811 (1998).
 - [17] Y. Hattori, R. Rubinstein, and A. Ishizawa, Shell model for rotating turbulence, *Phys. Rev. E* **70**, 046311 (2004).
 - [18] Jiang Mingshun and Liu Shida, Scaling behavior of velocity and temperature in a shell model for thermal convective turbulence, *Phys. Rev. E* **56**, 441 (1997).
 - [19] D. H. Wacks and C. F. Barenghi, Shell model of superfluid turbulence, *Phys. Rev. B* **84**, 184505 (2011).
 - [20] F. Plunian, R. Stepanov, and P. Frick, Shell models of magnetohydrodynamic turbulence, *Phys. Rep.* **523**, 1 (2013).

- [21] R. Benzi, L. Biferale, R. M. Kerr, and E. Trovatore, Helical shell models for three-dimensional turbulence, *Phys. Rev. E* **53**, 3541 (1996).
- [22] M. H. Jensen, G. Paladin, and A. Vulpiani, Shell model for turbulent advection of passive-scalar fields, *Phys. Rev. A* **45**, 7214 (1992).
- [23] A. A. Mailybaev, Spontaneously stochastic solutions in one-dimensional inviscid systems, *Nonlinearity* **29**, 2238 (2016).
- [24] D. Bandak, A. A. Mailybaev, G. L. Eyink, and N. Goldenfeld, Spontaneous stochasticity amplifies even thermal noise to the largest scales of turbulence in a few eddy turnover times, *Phys. Rev. Lett.* **132**, 104002 (2024).
- [25] P. Constantin, B. Levant, and E. S. Titi, Regularity of inviscid shell models of turbulence, *Phys. Rev. E* **75**, 016304 (2007).
- [26] I. Daumont, T. Dombre, and J.-L. Gilson, Instanton calculus in shell models of turbulence, *Phys. Rev. E* **62**, 3592 (2000).
- [27] R. Vinuesa and S. L. Brunton, Enhancing computational fluid dynamics with machine learning, *Nat. Comput. Sci.* **2**, 358 (2022).
- [28] C. Cho, J. Park, and H. Choi, A recursive neural-network-based subgrid-scale model for large eddy simulation: Application to homogeneous isotropic turbulence, *J. Fluid Mech.* **1000**, A76 (2024).
- [29] K. Duraisamy, Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence, *Phys. Rev. Fluids* **6**, 050504 (2021).
- [30] L. Biferale, A. A. Mailybaev, and G. Parisi, Optimal subgrid scheme for shell models of turbulence, *Phys. Rev. E* **95**, 043108 (2017).
- [31] G. Ortali, A. Corbetta, G. Rozza, and F. Toschi, Numerical proof of shell model turbulence closure, *Phys. Rev. Fluids* **7**, L082401 (2022).
- [32] J. Domingues Lemos and A. A. Mailybaev, Data-based approach for time-correlated closures of turbulence models, *Phys. Rev. E* **109**, 025101 (2024).
- [33] K. Um, R. Brand, Y. Fei, P. Holl, and N. Thuerey, Solver-in-the-loop: Learning from differentiable physics to interact with iterative PDE-solvers, *Advances in Neural Information Processing Systems* (2020), Vol. 33, pp. 6111–6122.
- [34] B. List, L.-W. Chen, K. Bali, and N. Thuerey, How temporal unrolling supports neural physics simulators, *Comput. Methods Appl. Mech. Eng.* **433**, 117441 (2025).
- [35] K. Fukushima, Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, *Biol. Cybern.* **36**, 193 (1980).
- [36] J. Sirignano, J. F. MacArt, and J. B. Freund, DPM: A deep learning PDE augmentation method with application to large-eddy simulation, *J. Comput. Phys.* **423**, 109811 (2020).
- [37] V. Shankar, V. Puri, R. Balakrishnan, R. Maulik, and V. Viswanathan, Differentiable physics-enabled closure modeling for Burgers’ turbulence, *Machine Learning: Sci. Technol.* **4**, 015017 (2023).
- [38] V. Shankara, D. Chakrabortya, V. Viswanathana, and R. Maulik, Differentiable turbulence: Closure as a PDE-constrained optimization, *Phys. Rev. Fluids* **10**, 024605 (2025).
- [39] B. Cheng and D. M. Titterton, Neural networks: A review from a statistical perspective, *Stat. Sci.* **9**, 2 (1994).
- [40] A. N. Kolmogorov, The local structure of turbulence in incompressible viscous fluid for very large Reynolds numbers, *Proc. Math. Phys. Eng. Sci* **434**, 9 (1991).
- [41] Z.-S. She and E. Leveque, Universal scaling laws in fully developed turbulence, *Phys. Rev. Lett.* **72**, 336 (1994).
- [42] S. Hochreiter and J. Schmidhuber, Long short-term memory, *Neural Comput.* **9**, 1735 (1997).
- [43] R. M. Larsen and T. Shpeisman, Tensorflow Graph Optimizations (2019), <https://research.google/pubs/tensorflow-graph-optimizations/>.