Computers and Fluids 311 (2026) 107028

Contents lists available at ScienceDirect

Computers and Fluids

journal homepage: www.elsevier.com/locate/compfluid

Energy-conserving neural network closure model for long-time accurate and stable 2D LES

T. van Gastelen [a,b,*], W. Edeling [a], B. Sanderse [a,b]

[a] Centrum Wiskunde & Informatica, Science Park 123, Amsterdam, the Netherlands
[b] Centre for Analysis, Scientific Computing and Applications, Eindhoven University of Technology, PO Box 513, Eindhoven, 5600 MB, the Netherlands

ARTICLE INFO

Keywords:
Large eddy simulation
Structure preservation
Closure modeling
Machine learning
Navier-Stokes equations

ABSTRACT

Machine learning-based closure models for large eddy simulation (LES) have shown promise in capturing complex turbulence dynamics but often suffer from instabilities and physical inconsistencies. In this work, we develop a novel skew-symmetric neural architecture as closure model that enforces stability while preserving key physical conservation laws. Our approach leverages a discretization that ensures mass, momentum, and energy conservation, along with a face-averaging filter to maintain mass conservation in coarse-grained velocity fields. We compare our model against several conventional data-driven closures (including unconstrained convolutional neural networks), and the physics-based Smagorinsky model. Performance is evaluated on decaying turbulence and Kolmogorov flow for multiple coarse-graining factors. In these test cases, we observe that unconstrained machine learning models suffer from numerical instabilities. In contrast, our skew-symmetric model remains stable across all tests, though at the cost of increased dissipation. Despite this trade-off, we demonstrate that our model still outperforms the Smagorinsky model in unseen scenarios. These findings highlight the potential of structure-preserving machine learning closures for reliable long-time LES.

1. Introduction

The incompressible Navier-Stokes equations are a set of partial differential equations (PDEs) that describe conservation of mass and momentum of fluid flows. They are used to model a multitude of flow phenomena, such as for the design of aircraft and ships, weather modeling, and even the formation of galaxies [1,2]. To simulate these phenomena, we solve the Navier-Stokes equations on a computational grid. This requires discretizing the differential operators present in the PDE. This can be done with various techniques, such as finite difference, finite volume, and finite element methods [3–5]. In this work we employ a structure-preserving finite difference scheme, introduced in [6]. The advantage of this scheme is that it not only satisfies mass and momentum conservation, but also conserves the global kinetic energy (in absence of viscosity and boundary contributions). We refer to conservation of mass, momentum and kinetic energy collectively as the physical 'structure' of the system, and we refer to such conservative discretization schemes as 'structure-preserving' or 'symmetry-preserving' [7,8]. Such schemes have the advantage of being unconditionally stable without relying on artificial diffusion, such as upwind schemes [9]. This makes them more suitable for long-time simulations, where correct physical energy behavior is crucial. However, problems arise when considering high Reynolds number flows [10]. For such flows we require very fine computational grids to resolve the smallest eddies in the system. This places a large (often insurmountable) burden on the computational resources.

A common approach to reduce computational requirements is large eddy simulation (LES). In LES, the system is coarse-grained by applying a filter to the velocity field, so that the dimension of the problem is effectively reduced. In this work, we take the 'discretize first, filter next' approach [11–13]. This means that coarse-graining is done on the discrete level by applying a discrete filter to a fine-grid discretization. In particular, we use a face-averaging filter, which has the advantage that the filtered velocity field still satisfies mass conservation [14]. The latter is necessary to preserve the energy-conserving properties of the convection operator. This provides substantial stability benefits, see [14]. From the coarse-graining procedure, a commutator error arises. In the 'discretize first, filter next' framework, this commutator error also includes the discretization error. Modeling the commutator error is referred to as closure modeling, and the corresponding models are closure models. Closure modeling is a main subject in LES research [10].

The most commonly used closure models are eddy-viscosity (functional) models [15,16]. Their primary job is to remove (dissipate) energy from the system to account for energy transfer from the resolved scales to the unresolved scales. These models approximate the subgrid stress

* Corresponding author.
E-mail addresses: tobyvangastelen@gmail.com (T. van Gastelen), w.edeling@cwi.nl (W. Edeling), B.sanderse@cwi.nl (B. Sanderse).

https://doi.org/10.1016/j.compfluid.2026.107028
Received 1 April 2025; Received in revised form 2 December 2025; Accepted 28 February 2026
Available online 4 March 2026

solely from the resolved scales by assuming proportionality between the subgrid-scale stress tensor and the rate-of-strain tensor [10,17]. The classical Smagorinsky model is the best-known example, but it is strictly dissipative [10,16,17] and therefore cannot represent backscatter, i.e., energy transfer from unresolved to resolved scales. Backscatter, however, plays an important role in many flows, including geophysical turbulence relevant for weather and climate modeling [18–20].

To address this limitation, structural models, such as scale-similarity closures, aim to reproduce the actual structure of the subgrid-scale stress tensor rather than parameterize its dissipation. While these models can, in principle, capture backscatter, they lack inherent dissipation. Consequently, they are often numerically unstable, requiring stabilization procedures such as explicit filtering or averaging. This need for both physical fidelity and numerical robustness motivated the development of mixed models, which combine a structural component with a dissipative functional component to achieve the accuracy of the former and the stability of the latter [21]. The dynamic Smagorinsky model [22] represents a related effort to increase flexibility within functional closures by adapting the eddy-viscosity coefficient based on resolved-scale information. Although it can produce limited backscatter, it too can suffer from numerical instabilities [23]. Several physics-based strategies have since been proposed to improve backscatter representation while maintaining stability. These include enforcing a consistent subgrid-scale energy budget [24], introducing filter-based artificial dissipation [25], and imposing explicit dissipation constraints [26,27]. Nevertheless, state-of-the-art subgrid-scale closures used in global climate models still exclude backscatter [28], primarily due to concerns over robustness. In response, the machine-learned closure models that we will propose aim, in a somewhat similar spirit as mixed models, to combine the best of functional and structural approaches: we keep the structure of the subgrid stress tensor, while at the same time ensuring that the models are energy dissipative (but not as much as existing functional models).

Shifting our attention to such machine learning algorithms, neural networks have shown great potential in accurately modeling the closure term, while accounting for backscatter [15,29–36]. Offline testing often shows good agreement of the neural network outputs with the true closure term. However, when using the closure term in an actual simulation, problems arise and instabilities occur [29,31,32,34,36,37]. One approach to handle this issue is by clipping the neural network such that the output becomes strictly dissipative, for example by projecting onto an eddy-viscosity basis [34,36,37]. However, this results in closure models which are generally too dissipative [36], which will be confirmed by our results. Another way to deal with instabilities is to add artificial noise to the training [38]. However, in [38] it was shown that this only delays the instability and does not prevent it entirely. Stochastic approaches have also been suggested, e.g. [35] combines idealized LES with neural stochastic differential equations and show increased stability.

Stability issues are often combatted by introducing some form of *a posteriori* learning [11,12,30,31,39–42]. In this way, a closure model is trained based on reproducing the solution trajectory rather than reproducing the closure term itself. This aids in the stability of the LES. In [31], it was shown that by increasing the number of solver steps one unrolls during training, the stability and performance are improved. Melchers et al. [12] shows that there is an optimum in the number of unrolled steps, related to the chaotic nature of the system. However, in [36] it is shown that limited training data still causes instabilities for neural network-based closure models.

In [43], a strictly local small neural network approach is suggested for the representation of the subrid-scale stress tensor. Here, a small set of carefully chosen Galilean invariant and non-dimensionalized inputs are used such that the resulting closure model is symmetric, Galilean invariant, rotationally and reflectionally invariant, and unit invariant. The authors show that training on a single snapshot is enough to obtain a closure model which generalizes well to their considered test cases, even without clipping. However, in [15] it is shown that convolutional neural networks (CNNs), combined with Fourier neural operators, with non-invariant inputs, is capable of outperforming such approaches. For an overview on machine learning-based closure modeling see [42].

However, while the aforementioned references observed improved stability, none of the discussed approaches *guarantees* stability, without applying some form of backscatter clipping, or other ad-hoc measures, such as e.g. trial-and-error by changing input features [34] or data augmentation [38]. This makes long-time LES with a data-driven closure model unreliable. Given that we are especially interested in the statistics of turbulent flows, e.g. the average energy spectrum obtained over long time horizons, it is of crucial importance that the combination of discretization and closure model yields stable simulations. Hence, the main aim and novelty of this article is to derive a machine-learned LES closure model, where stability is guaranteed by design, independent of the network weights or amount of training data. To achieve this, we propose to build upon our earlier work presented in [13]. In this work, the closure model was represented by an energy-conserving skew-symmetric term and a dissipative symmetric negative-definite term. This was accomplished by using a set of compressed subgrid-scale variables that allow the transfer of energy from unresolved scales to resolved scales. In this work, we extend this approach from one to two spatial dimensions. While it is certainly true that turbulence is fundamentally a 3D phenomenon, we reiterate that the main aim of our article is focused on the stability issue, which has largely been studied in the context of 2D turbulence, see [15,29,31–33,36,39] among others.

As a first step, we exclude the compressed subgrid variables, as their precise meaning and definition in multiple dimensions is still an open question. Instead, we show that the skew-symmetric framework without subgrid variables already offers significant stability advantages over existing approaches. Although this means we do not explicitly account for backscatter, the skew-symmetric term is still capable of redistributing energy throughout the domain. This extends the predictive capability of the closure model beyond an eddy-viscosity basis. The work presented in [44] discusses similar ideas regarding structure-preserving neural networks, although outside the realm of LES.

The outline of this paper is as follows: In Section 2, we start with introducing the incompressible Navier-Stokes equations, the physical structure present in the system, and the structure-preserving discretization. In Section 3, we discuss coarse-graining of the discrete system of equations by applying a discrete spatial filter, we derive the exact closure term, and discuss closure modeling approaches. In Section 4, we introduce the machine learning-based closure models: using a CNN to model the closure term, using a CNN to model the subgrid-scale stress tensor, and finally our skew-symmetric neural network architecture. An extended motivation for this architecture can be found in D. Next, we discuss the test cases and the results in Section 5, regarding closure model performance and stability. We conclude our work in Section 6.

2. Preliminaries

2.1. Navier-Stokes equations

In convervative form, the incompressible Navier-Stokes equations read as follows:

$$\frac{\partial \mathbf{u}}{\partial t} + \nabla \cdot (\mathbf{u}\mathbf{u}^T) = -\nabla p + \nu \nabla^2 \mathbf{u} + \mathbf{f}, \tag{1a}$$

$$\nabla \cdot \mathbf{u} = 0. \tag{1b}$$

This PDE describes the evolution of a fluid velocity field $\mathbf{u}(\mathbf{x}, t) \in \mathbb{R}^d$ in d-dimensional space $\mathbf{x} \in \mathbb{R}^d$ and time t. Here we restrict ourselves to 2D such that $\mathbf{u}(\mathbf{x}, t) = \begin{bmatrix} u(\mathbf{x}, t) & v(\mathbf{x}, t) \end{bmatrix}^T$. The different forces acting on the velocity field are due to convection, the gradient of the pressure $p(\mathbf{x}, t)$, and friction (for a nonzero viscosity $\nu \geq 0$), appearing from left to right in the equation. In addition, there is $\mathbf{f}(\mathbf{x}, t) \in \mathbb{R}^2$ which represents

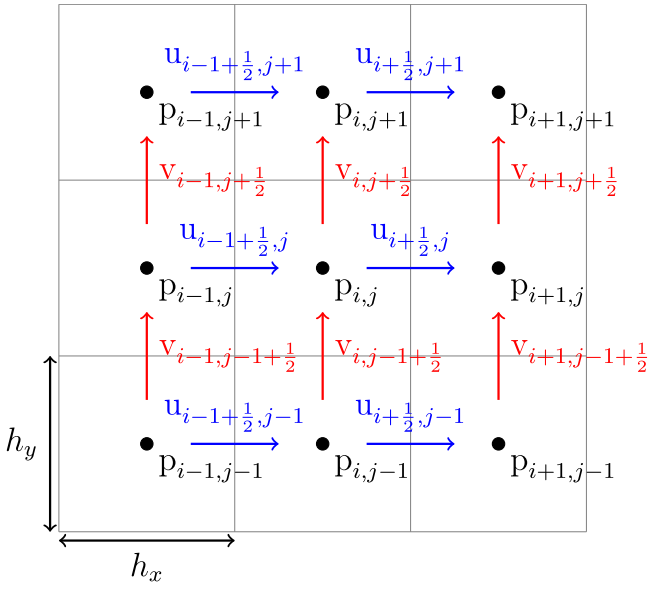

Fig. 1. Staggered grid discretization of Navier-Stokes equations. The pressure points are indexed with whole numbers, while for the velocity components we have an offset of $\frac{1}{2}$ in the appropriate direction.

body-forces, such as gravity. In this text we refrain from discussing boundary conditions, as we consider a periodic spatial domain Ω.

2.2. Physical structure

This set of equations represent a set of fundamental physical laws, namely: conservation of mass, momentum $\mathbf{P} = \int_\Omega \mathbf{u} \mathrm{d}\Omega$, and (kinetic) energy $E := \frac{1}{2}\int_\Omega \mathbf{u}\cdot\mathbf{u}\mathrm{d}\Omega$ (for $\nu = 0$). These are collectively referred to as the system's physical structure. In equation form, these laws read

$$\nabla \cdot \mathbf{u} = 0, \tag{2}$$

$$\frac{\mathrm{d}\mathbf{P}}{\mathrm{d}t} = \int_\Omega \frac{\partial \mathbf{u}}{\partial t}\mathrm{d}\Omega = \int_\Omega \mathbf{f}(x,t)\mathrm{d}\Omega, \tag{3}$$

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \int_\Omega \mathbf{u}\cdot\frac{\partial \mathbf{u}}{\partial t}\mathrm{d}\Omega = -\int_\Omega \nu||\nabla\mathbf{u}||_2^2\mathrm{d}\Omega + \int_\Omega \mathbf{u}\cdot\mathbf{f}(\mathbf{x},t)\mathrm{d}\Omega, \tag{4}$$

Derivations of these conservation laws are presented in Appendix A. From these laws, we can see that momentum and energy (for zero dissipation) are conserved in the absence of forcing.

2.3. Discretization

To simulate practical use cases, we require discretizing the set of Eq. (1a) on a computational grid. Here we employ a structure-preserving second-order accurate finite difference discretization on a staggered grid [3,6]. This discretization is chosen as it satisfies the energy-conserving properties of the convective term. The employed discretization consists of in total $N_x \times N_y = N$ cells Ω_{ij} with $i = 1,\dots,N_x$ and $i = 1,\dots,N_y$ for a 2D flow case. The semi-discrete set of equations are written as

$$\mathbf{\Omega}_h \frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} + \mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h = -\mathbf{G}_h\mathbf{p}_h + \nu\mathbf{D}\mathbf{u}_h + \mathbf{\Omega}_h\mathbf{f}_h, \tag{5a}$$

$$\mathbf{M}_h\mathbf{u}_h = \mathbf{0}, \tag{5b}$$

where $\mathbf{u}_h \in \mathbb{R}^{2N}$ contains the approximations of u and v on the cell faces and $\mathbf{p}_h \in \mathbb{R}^N$ the approximation of p in the cell centers. A schematic representation of the employed uniform staggered grid is displayed in Fig. 1. The grid-spacing in each direction is indicated by h_x and h_y.

Furthermore, $\mathbf{\Omega}_h$ contains the cell volumes on the diagonal. The operators in (1a) are now represented by matrices. $\mathbf{C}_h(\mathbf{u}_h) \in \mathbb{R}^{2N\times 2N}$ represents the convection operator, $\mathbf{G}_h \in \mathbb{R}^{2N\times N}$ the gradient operator, $\mathbf{D}_h \in \mathbb{R}^{2N\times 2N}$ the diffusion operator, $\mathbf{M}_h \in \mathbb{R}^{N\times 2N}$ the divergence operator, and $\mathbf{f}_h \in \mathbb{R}^{2N}$ the forcing at the cell faces.

2.4. Structure of the discretization

This discretization preserves the physical structure in a discrete sense. Discretely, the total momentum and energy are approximated as

$$\mathbf{P}_h = \underbrace{\begin{bmatrix}\mathbf{1}_h & \mathbf{0}_h \\ \mathbf{0}_h & \mathbf{1}_h\end{bmatrix}^T}_{=:\mathbb{1}_h} \mathbf{\Omega}_h\mathbf{u}_h, \tag{6}$$

$$E_h = \frac{1}{2}\mathbf{u}_h^T\mathbf{\Omega}_h\mathbf{u}_h, \tag{7}$$

where $\mathbf{0}_h, \mathbf{1}_h \in \mathbb{R}^N$ are column vectors of zeros and ones, respectively. The change of these quantities for this discretization are given by

$$\frac{\mathrm{d}\mathbf{P}_h}{\mathrm{d}t} = \mathbb{1}_h\frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} = \mathbb{1}_h\mathbf{\Omega}_h\mathbf{f}_h, \tag{8}$$

$$\frac{\mathrm{d}E_h}{\mathrm{d}t} = \mathbf{u}_h^T\frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} = -\nu||\mathbf{Q}_h\mathbf{u}_h||_2^2 + \mathbf{u}_h^T\mathbf{\Omega}_h\mathbf{f}_h, \tag{9}$$

where we used the fact that the diffusion operator $\mathbf{D}_h$ can be Cholesky decomposed as $-\mathbf{Q}_h^T\mathbf{Q}_h$ [45]. As the discrete energy is solely decreasing, in the absence of forcing, this discretization provides stability. The convective contribution disappears due to the skew-symmetry of the discrete operator, however this requires the discrete solution $\mathbf{u}_h$ to be divergence free, i.e., $\mathbf{M}_h\mathbf{u}_h = \mathbf{0}_h$. A more elaborate discussion is presented in Appendix B.

2.5. Pressure projection

The divergence freeness can also be written as a projection of the right-hand side (RHS) of the PDE discretization (5) on a divergence-free basis. We introduce this formulation of the discrete system because it is more convenient for closure modeling [14], as it combines (5a) and (5b) into a single equation. The projection looks as follows:

$$\mathbf{\Omega}_h\frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} = \mathcal{P}_h\mathbf{m}_h(\mathbf{u}_h), \tag{10}$$

where $\mathbf{m}_h(\mathbf{u}_h) \in \mathbb{R}^{2N}$ contains the operators on the RHS of (5a), i.e.,

$$\mathbf{m}_h(\mathbf{u}_h) = -\mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h + \nu\mathbf{D}_h\mathbf{u}_h + \mathbf{\Omega}_h\mathbf{f}_h, \tag{11}$$

and $\mathcal{P}_h \in \mathbb{R}^{2N\times 2N}$ projects the RHS on a divergence free basis. $\mathcal{P}_h$ is defined as

$$\mathcal{P}_h := (\mathbf{I} - \mathbf{G}_h(\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{G}_h)^{-1}\mathbf{M}_h\mathbf{\Omega}_h^{-1}). \tag{12}$$

A derivation of this operator is presented in Appendix C.

3. Closure modeling

3.1. Filtering

As stated earlier, carrying out a direct numerical simulation (DNS) is often infeasible for practical use cases. This is why we aim to solve a coarse-grained set of equations. Coarse-graining is done by applying a filter to the velocity field. In our case, we take the 'discretize first, filter next' approach [11,12]. This means we start by discretizing our velocity field on an adequately fine grid, such that we resolve the relevant scales of the flow. Next, we apply a linear filter $\mathbf{W}^{2\bar{N}\times 2N}$ to obtain the filtered velocity field:

$$\bar{\mathbf{u}}_H = \mathbf{W}\mathbf{u}_h, \tag{13}$$

where the filtered velocity $\bar{\mathbf{u}}_H \in \mathbb{R}^{2\bar{N}}$ lives on a coarse grid consisting of $\bar{N} = \bar{N}_x \times \bar{N}_y$ grid cells. In our case we employ a face-averaging filter, as suggested in [14]. A schematic representation of the filter is shown in Fig. 2.

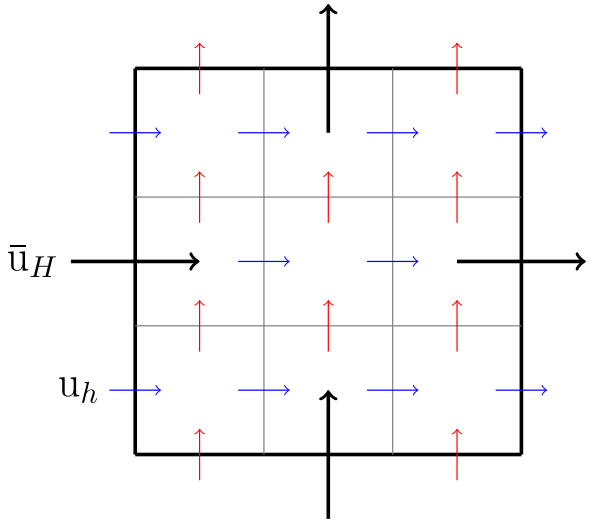

Fig. 2. Schematic representation of the face-averaging filter. In this example, a single coarse-grained cell contains nine fine-grid cells. Three fine-grid velocity components in $\mathbf{u}_h$ on each of the coarse cell faces are averaged to obtain the filtered velocity field $\bar{\mathbf{u}}_H$.

The advantage of this filter, as opposed to, for example, a volume-averaging filter, is that the filtered velocity satisfies divergence freeness on the coarse grid. This ensures skew-symmetry of the coarse-grid convection operator, which aids in stability of the coarse-grained system of equations [14].

3.2. System of equations

To model the behavior of the filtered velocity field, we take the following ansatz:

$$\mathbf{\Omega}_H \frac{\mathrm{d}\bar{\mathbf{u}}_H}{\mathrm{d}t} \approx \mathcal{P}_H \mathbf{m}_H(\bar{\mathbf{u}}_H) + \tilde{\mathbf{c}}(\bar{\mathbf{u}}_H, \theta), \tag{14}$$

where the subscript H indicates a coarse-grid equivalent to the operators discussed in Section 2.3 and $\tilde{\mathbf{c}}(\bar{\mathbf{u}}_H, \theta)$ is a (neural network-based) closure model with parameters θ. The projection on a divergence-free basis is achieved through $\mathcal{P}_H$, which is justified due to the face-averaging filter. The closure model is required as the coarse discretization does not resolve all the relevant scales of the flow. In addition, the coarse grid results in a discretization error. Both of these are captured in the commutator error with respect to the fine-grid discretization. The true evolution of $\bar{\mathbf{u}}_H$ is given by

$$\mathbf{\Omega}_H \frac{\mathrm{d}\bar{\mathbf{u}}_H}{\mathrm{d}t} = \mathcal{P}_H \mathbf{m}_H(\bar{\mathbf{u}}_H) + \underbrace{(\mathbf{W}\mathcal{P}_h \mathbf{m}_h(\mathbf{u}_h) - \mathcal{P}_H \mathbf{m}_H(\bar{\mathbf{u}}_H))}_{=:\mathbf{c}(\mathbf{u}_h)}, \tag{15}$$

where the commutator error $\mathbf{c}_h(\mathbf{u}_h)$ includes both sources of error. As the true filtered velocity field is divergence-free on the coarse grid, we include the closure model in the projection to preserve divergence freeness, as suggested by [14]. This changes ansatz (14) to

$$\mathbf{\Omega}_H \frac{\mathrm{d}\bar{\mathbf{u}}_H}{\mathrm{d}t} \approx \mathcal{P}_H(\mathbf{m}_H(\bar{\mathbf{u}}_H) + \tilde{\mathbf{c}}(\bar{\mathbf{u}}_H, \theta)). \tag{16}$$

The energy contribution of the closure model is computed as

$$\text{energy contribution} = \bar{\mathbf{u}}_H^T \tilde{\mathbf{c}}(\bar{\mathbf{u}}_H, \theta), \tag{17}$$

where the resolved energy is given by

$$\bar{E}_H = \frac{1}{2} \bar{\mathbf{u}}_H^T \mathbf{\Omega}_H \bar{\mathbf{u}}_H. \tag{18}$$

In addition, the total momentum $\bar{\mathbf{P}}_H = \mathbb{1}_H \mathbf{\Omega}_H \mathbf{u}_H$ is conserved if

$$\mathbb{1}_H \tilde{\mathbf{c}}(\mathbf{u}_H, \theta) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{19}$$

holds for the closure model.

3.3. Energy analysis of the closure term

Based on training data, we analyze the energy contribution of the true closure term, see (17), for different levels of coarse-graining. In this case, the reference simulation was carried out on a 2048×2048 grid. Exact simulation conditions are discussed in Section 5.1. The resulting energy contributions, in addition to the resolved energy trajectories, are presented in Fig. 3.

We observe that for a resolution of 32×32 the closure term produces a significant amount of backscatter, as the energy contribution is mostly positive. Also, in the resolved energy trajectory we observe that the decay is less smooth, as opposed to larger resolutions. This means dissipative models will likely perform poorly for this resolution. For a resolution of 64×64, we find the energy contribution to be mostly dissipative, whereas for 128×128 it is strictly dissipative for this test case. This motivates the use of dissipative closure models, such as the skew-symmetric neural network architecture we will introduce in Section 4.3

3.4. Fitting of the model parameters

To find the optimal set of parameters θ, one can take different approaches. The most straightforward approach would be to match the true commutator error as best as possible. However, this often results in inaccurate simulations or even instabilities [11,12,29–31,37,38]. This is why we resort to optimizing the parameters in order to accurately reproduce the filtered direct numerical simulation (FDNS) solution, also known as 'trajectory fitting' or 'solver in the loop'. The corresponding loss function is:

$$\mathcal{L}_n(\mathbf{X}; \theta) = \sum_{\mathbf{u}_h \in \mathbf{X}} \sum_{i=1}^{n} ||\bar{S}_\theta^i(\mathbf{W}\mathbf{u}_h) - \mathbf{W} S^{i(\overline{\Delta t}/\Delta t)}(\mathbf{u}_h)||_2^2, \tag{20}$$

where $\mathbf{X}$ is a snapshot matrix consisting of samples of $\mathbf{u}_h$ as columns, representing the training data set. The notation $\bar{S}_\theta^i(\mathbf{W}\mathbf{u}_h)$ represents the predicted coarsened velocity field after applying an explicit time integration scheme for i steps, each with a step size of $\overline{\Delta t}$, starting from the initial condition $\mathbf{W}\mathbf{u}_h$, incorporating the closure model. The corresponding DNS solution is denoted by $S^{i(\overline{\Delta t}/\Delta t)}(\mathbf{u}_h)$, using a smaller step size Δt and initialized with $\mathbf{u}_h$. The ratio $\overline{\Delta t}/\Delta t$ arises because the coarse grid permits larger time steps [46]. Note that $\bar{S}_\theta$ is the solver that works on the coarse grid and incorporates the closure model, while S is the DNS solver that works on the fine grid. The Adam optimization algorithm [47] will be used to minimize the loss. Further details on the training procedure are discussed in Section 5.1.

4. Methodology

As explained in Section 3, a main challenge in closure modeling is deriving an expression for $\tilde{\mathbf{c}}(\bar{\mathbf{u}}_H, \theta)$. In this section, we propose a skew-symmetric framework that results in a new expression for $\tilde{\mathbf{c}}(\bar{\mathbf{u}}_H, \theta)$, providing stability. In Sections 4.1 and 4.2 we first introduce the Smagorinsky model and CNNs, respectively. These not only serve as something to compare our framework to, but also as necessary preliminaries. In Section 4.3 the new framework is derived.

4.1. Smagorinsky model

We start off with the Smagorinsky model, which is an eddy-viscosity model. As stated earlier, the main assumption in eddy-viscosity models is that this subgrid-scale stress tensor is proportional to the rate-of-strain tensor $\bar{\mathbf{S}}_{ij} = \frac{1}{2}(\frac{\partial \bar{\mathbf{u}}_i}{\partial \mathbf{x}_j} + \frac{\partial \bar{\mathbf{u}}_j}{\partial \mathbf{x}_i})$, where $\bar{\mathbf{u}} \in \mathbb{R}^2$ is a continuous representation of the resolved velocity field. Eddy-viscosity type closure models have the following form:

$$\tilde{c}(\bar{\mathbf{u}}) = \nabla \cdot (\nu_t \bar{\mathbf{S}}), \tag{21}$$

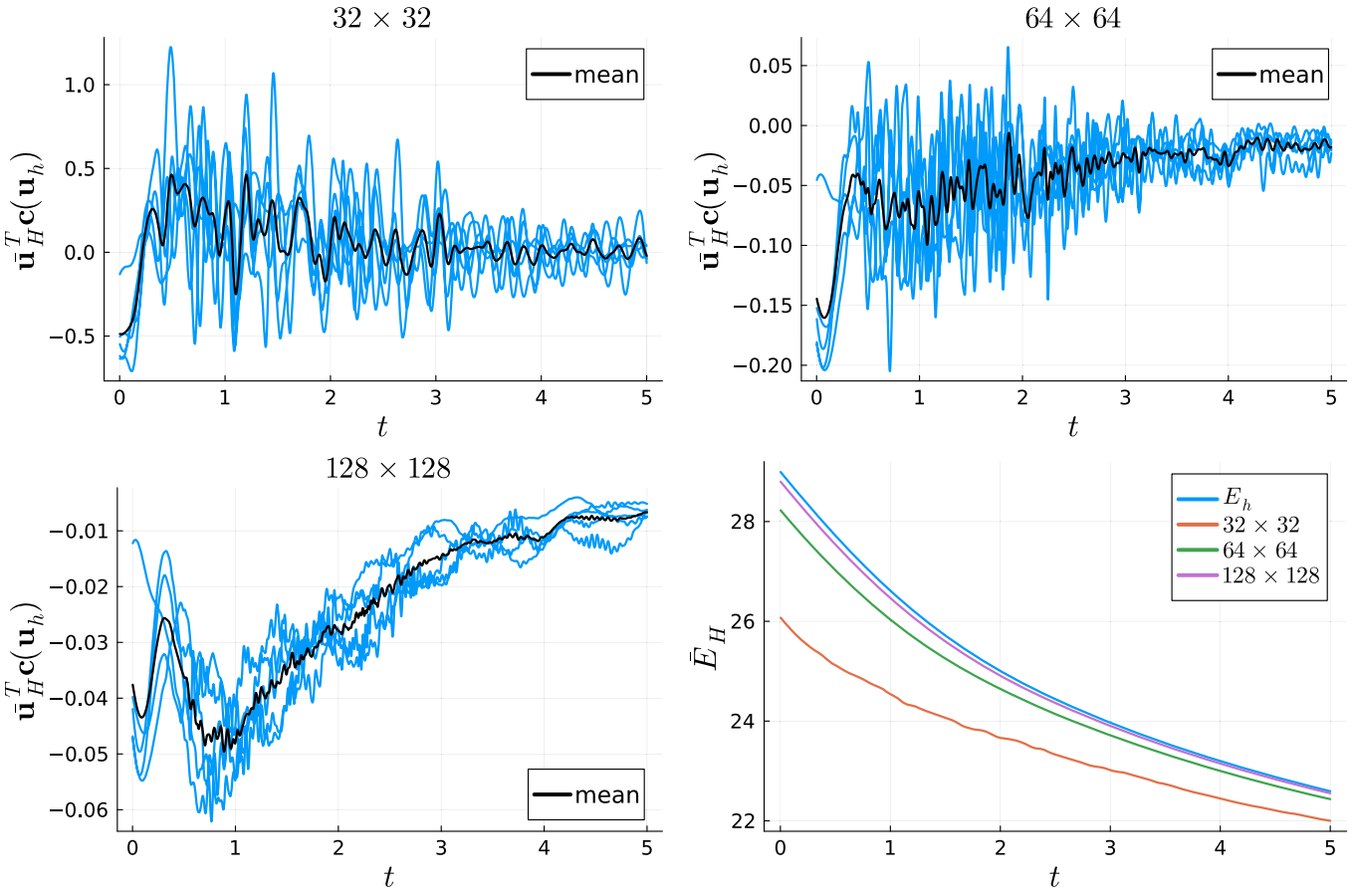

Fig. 3. Resolved energy contributions for the true closure term, computed for five decaying turbulence simulations which constitute the training data for the machine learning closure models. The trajectories are presented for different levels of coarse-graining. The reference grid has a resolution of 2048×2048. See Section 5.1 for the exact simulation conditions. (Bottom-right) Resolved energy trajectories for one of the simulations.

where $\nu_t \geq 0$ is the eddy-viscosity. The Smagorinsky model assumes the following form for ν_t:

$$\nu_t = (C_s \Delta)^2 \sqrt{2\mathrm{tr}(\bar{\mathbf{S}}^2)}, \tag{22}$$

where Δ is often chosen as the grid-spacing, i.e. $\Delta = \sqrt{h_x h_y}$. The model contains only a single parameter C_s, which can be tuned. In practice, we use a discretization of the Smagorinsky model. This yields a discrete closure model:

$$\tilde{\mathbf{c}}^{\mathrm{SMAG}}(\bar{\mathbf{u}}_H, C_s) = (C_s \Delta)^2 \sqrt{2\mathrm{tr}(\bar{\mathbf{S}}_H^2)} \nabla_H \cdot \bar{\mathbf{S}}_H, \tag{23}$$

where the subscript H indicates a discretization of the derivative operators. We employ a central difference scheme for these derivatives. The global energy contribution, see (17), of the Smagorinsky model is always negative, i.e., it is strictly dissipative. In addition, the momentum conservation constraint (19) is satisfied for the Smagorinsky model.

4.2. Neural network closure

A straightforward machine learning approach is to use a CNN as a closure model, i.e.

$$\tilde{\mathbf{c}}^{\mathrm{CNN}}(\bar{\mathbf{u}}_H, \theta) = \mathrm{CNN}(\bar{\mathbf{u}}_H, \theta). \tag{24}$$

These models are well-suited to Cartesian grids [14,31,33], such as the one we employ here. In addition, they are translation equivariant. Here a CNN is used to map the filtered solution $\bar{\mathbf{u}}_H \in \mathbb{R}^{2\tilde{N}}$ to $\tilde{\mathbf{c}} \in \mathbb{R}^{2\tilde{N}}$ by chaining a series of convolutions with non-linear activation functions σ^n, where n indicates which layer of the CNN is considered. A single layer of such a network is represented as

$$\mathbf{z}^{n+1} = \sigma^n(\mathcal{A}^n \mathbf{z}^n + \mathbf{b}^n), \tag{25}$$

where each vector $\mathbf{z}^n$ contains a set of fields, typically referred to as 'channels' in CNN literature. The matrix $\mathcal{A}$ contains $s_{n+1} \times s_n$ submatrices encoding convolutional stencils, where s_n is the number of channels represented in $\mathbf{z}_n$. By choosing $s_{n+1} > s_n$, the data is effectively lifted to a higher-dimensional space. The vector $\mathbf{b}_n$ contains s_{n+1} bias channels, which are constant fields each determined by a single parameter. For example, a single convolution $\mathbf{A}$ (parameterized by weights α_{jk}) of a single channel $\mathbf{z}$, with bias vector $\mathbf{b}$ (parameterized by bias β), is represented as

$$(\mathbf{A}\mathbf{z} + \mathbf{b})_{ij} = \sum_{k,l=-r}^{r} (\alpha_{kl} \mathbf{z}_{i+k,j+l}) + \beta \tag{26}$$

The double index notation is used to indicate the location on the 2D grid (one index for each spatial dimension), see Fig. 1. Moreover, r represents the radius of the convolution in both spatial directions. On the edge of the domain, one uses padding of $\mathbf{z}$ to keep the size of the channels constant. In our case, we use circular padding to represent periodic boundary conditions (BCs). For intermediate layers, we use a ReLU activation function σ^n, and for the final layer, we take σ^n to be the identity [48]. Using a CNN as a closure model does allow for modeling backscatter as its energy contribution, see (17), can be both negative and positive. However, it violates momentum conservation, see (19).

A straightforward way to resolve the latter is to use a CNN to predict a stress tensor $\boldsymbol{\tau}^{\mathrm{CNN}}$. The closure model is then obtained by taking the

divergence of this tensor [12,34]:

$$\tilde{\mathbf{c}}^{\mathrm{DIV}}(\bar{\mathbf{u}}_H,\theta) = \nabla_H \cdot \boldsymbol{\tau}^{\mathrm{CNN}}(\bar{\mathbf{u}}_H,\theta). \tag{27}$$

This ensures that momentum conservation, see (19), is satisfied. This formulation will be referred to as DIV. However, neither of these formulations is guaranteed to be stable and can therefore result in poor simulation results.

4.3. Skew-symmetric framework

For our novel closure modeling approach we build on earlier work which we presented in [13]. In this work a skew-symmetric neural network architecture was introduced and applied to 1D equations. Moreover, a coarse-grid representation of the subgrid-scale energy was included in the coarse-grained system. However, as stated earlier in Section 1, it is unclear how to extend this representation to multiple dimensions. On the other hand, applying the proposed neural network architecture, without this subgrid-scale representation, to 2D problem does not require any modifications to the architecture.

In this skew-symmetric framework, the closure model is represented as the sum of a skew-symmetric and negative-definite term:

$$\tilde{\mathbf{c}}^{\mathrm{SKEW}}(\bar{\mathbf{u}}_H,\theta) = (\mathcal{K}-\mathcal{K}^T)\bar{\mathbf{u}}_H - \mathcal{Q}^T\mathcal{Q}\bar{\mathbf{u}}_H, \tag{28}$$

where $\mathcal{K}(\bar{\mathbf{u}}_H,\theta), \mathcal{Q}(\bar{\mathbf{u}}_H,\theta) \in \mathbb{R}^{2\tilde{N}\times 2\tilde{N}}$ are build from CNN outputs. The different terms in this closure model are represented by matrix multiplications of the velocity vector. Model flexibility stems from the fact that these matrices depend nonlinearly on the solution vector via neural network outputs. The closure model is always dissipative:

$$\bar{\mathbf{u}}_H^T\tilde{\mathbf{c}}^{\mathrm{SKEW}}(\bar{\mathbf{u}}_H,\theta) = \bar{\mathbf{u}}_H^T(\mathcal{K}-\mathcal{K}^T)\bar{\mathbf{u}}_H - \bar{\mathbf{u}}_H^T\mathcal{Q}^T\mathcal{Q}\bar{\mathbf{u}}_H = -||\mathcal{Q}\bar{\mathbf{u}}_H||_2^2 \leq 0, \tag{29}$$

as the skew-symmetric contribution cancels. This means the closure model is guaranteed to be stable, unlike the previously presented machine learning approaches. Note that stability is only guaranteed up to a time discretization error. Analyzing stability while accounting for the time discretization error would require knowledge of the Jacobian of the system. This makes formal analysis rather cumbersome, due to nonlinearity of the neural network [49]. Therefore, we consider this outside the scope of this research.

To ensure stability, ignoring the time discretization, the closure model does not allow for backscatter, but does increase modeling freedom beyond an eddy-viscosity basis through the skew-symmetric term. To see this, we note any real square matrix $\mathbf{A}$ can be decomposed uniquely into the sum of a symmetric and a skew-symmetric matrix [50,51]:

$$\mathbf{A} = \mathbf{S} + \mathbf{R}, \qquad \mathbf{S} = \tfrac{1}{2}(\mathbf{A}+\mathbf{A}^T), \quad \mathbf{R} = \tfrac{1}{2}(\mathbf{A}-\mathbf{A}^T). \tag{30}$$

By constraining the symmetric part $\mathbf{S}$ to be negative definite, we restrict ourselves to the space of stable linear operators, since for any nonzero $\mathbf{x}$,

$$\mathbf{x}^T\mathbf{S}\mathbf{x} < 0 \quad \Rightarrow \quad \mathbf{x}^T\mathbf{A}\mathbf{x} = \mathbf{x}^T\mathbf{S}\mathbf{x} < 0, \tag{31}$$

implying strictly decreasing energy in all directions. Diagonalizing the symmetric operator $\mathbf{S}$ yields real, strictly negative eigenvalues and an orthogonal eigenbasis. Hence, the system's energy decays independently along these orthogonal modes. This negative-definite term can thus be interpreted as a generalized eddy-viscosity model that captures dissipative effects. However, a purely negative-definite formulation cannot represent conservative or rotational behaviors, since those correspond to energy-preserving dynamics for which $\mathbf{x}^T\mathbf{A}\mathbf{x} = 0$. The skew-symmetric component $\mathbf{R}$ addresses this limitation: it represents interactions that conserve energy, such as rotations, couplings, or advective terms in fluid dynamics [52]. By combining both components, we obtain the representation

$$\mathbf{A} = (\mathbf{K}-\mathbf{K}^T) - \mathbf{Q}^T\mathbf{Q}, \tag{32}$$

which spans the entire space of stable linear operators while remaining expressive enough to capture both dissipative and conservative effects. This serves as motivation for the proposed decomposition of the neural network architecture. This decomposition is not unique, since $\mathbf{A}$ remains the same if $\mathbf{K}$ is perturbed by an arbitrary symmetric matrix.

During our testing, we observed the effect of the skew-symmetric term to be much larger than that of the negative-definite term, see Fig. 8. This further motivates its presence. From the computed energy contributions of the true closure term (see Fig. 3), we believe backscatter is limited for reasonable coarse-graining factors. This means such a constrained closure model should be able to capture the energy behavior of the true closure term. In the upcoming sections, we will explain how the operators in this skew-symmetric architecture are built.

4.3.1. Skew-symmetric term

As described in [13], $\mathcal{K}$ is constructed as follows:

$$\mathcal{K}(\bar{\mathbf{u}}_H,\theta) = \mathcal{B}_1^T\mathbf{k}(\bar{\mathbf{u}}_H,\theta)\mathcal{B}_2, \tag{33}$$

where $\mathbf{k}(\bar{\mathbf{u}}_H,\theta) = \mathrm{diag}(\mathbf{k}_1,\mathbf{k}_2) \in \mathbb{R}^{2\tilde{N}\times 2\tilde{N}}$ is a matrix containing the neural network outputs $\mathbf{k}_1,\mathbf{k}_2 \in \mathbb{R}^{2\tilde{N}}$ on the diagonal. Here we choose to represent each matrix in the decomposition to be square. However, one is free to choose the dimensions of the matrices, as long as the $\mathcal{K}$ is square. Also, the sparsity of the matrices can be varied. In our case, the underlying neural network architecture is a CNN, as we employ a uniform grid. This means a CNN is used to map $\bar{\mathbf{u}}_H$ to output channels $\mathbf{k}_1$ and $\mathbf{k}_2$. The matrices $\mathcal{B}$ are linear convolutional layers mapping from two input channels to two output channels, similarly to the convolutional layers introduced in Section 4.2. The $\mathcal{B}$ matrices thus contain four submatrices encoding convolutions. A clear motivation for this specific decomposition of $\mathcal{K}$, besides sparsity and computational efficiency, is outlined in Appendix D. The main idea is that the proposed decomposition resembles a simple advection operator for specific choices of $\mathcal{B}$ matrices, where the matrix $\mathbf{k}$ supplies enough degrees of freedom to freely traverse the energy-conserving solution space, by locally controlling the advection in each direction. Another option would be to use more output channels, i.e., $\mathbf{k}_1,\dots,\mathbf{k}_n$ and construct a larger matrix $\mathbf{k} = \mathrm{diag}(\mathbf{k}_1,\dots,\mathbf{k}_n)$, where n would be a hyperparameter. This would require the $\mathcal{B}$ matrices to be non-square and contain more submatrices encoding different convolutions. Such an extended architecture could possibly lead to faster convergence of the training procedure and more expressive power. In addition, a smaller $\mathbf{k}$ matrix, i.e. $\mathbf{k} \in \mathbb{R}^{c\times c}$ with $c << \tilde{N}$, could also be considered. This would effectively apply the nonlinear neural network operations in a reduced latent space, encoded by the $\mathcal{B}$ matrices [53]. We consider both options outside the scope of the current work, but view them as interesting future research directions.

To satisfy momentum conservation, see (19), we require the sum of the convolution weights in the submatrices to be zero such that both $\mathcal{B}$ and $\mathcal{B}^T$ are in the nullspace of $\mathbb{1}_H$. To achieve this, we let the weights $\bar{b}_{kl}$ of such a convolution depend on a set of parameters b_{kl}:

$$\bar{b}_{kl} = b_{kl} - \frac{1}{(2r+1)^2}\sum_{k,l=-r}^{r} b_{kl}, \tag{34}$$

such that $\sum_{k,l=-r}^{r}\bar{b}_{kl} = 0$ holds. To extend the architecture to unstructured grids, the convolutions in the CNN and in the $\mathcal{B}$ matrices can simply be replaced by graph convolutions.

4.3.2. Negative-definite term

As described in [13], $\mathcal{Q}$ is constructed as follows

$$\mathcal{Q}(\bar{\mathbf{u}}_H,\theta) = \mathbf{q}(\bar{\mathbf{u}}_H,\theta)\mathcal{B}_3, \tag{35}$$

where $\mathbf{q}(\bar{\mathbf{u}}_H,\theta) = \mathrm{diag}(\mathbf{q}_1,\mathbf{q}_2)$ is also constructed from outputs of the CNN $\mathbf{q}_1,\mathbf{q}_2 \in \mathbb{R}^{2\tilde{N}}$. This means the CNN has in total four output channels to build the fields $\mathbf{k}_1,\mathbf{k}_2,\mathbf{q}_1,\mathbf{q}_2$. The parameters of the model include both the CNN weights and the parameters of the $\mathcal{B}$ matrices. As these are all sparse operations, the model remains computationally efficient. Because the model only contains convolutions, it is translation equivariant. Furthermore, it can be safely applied to larger grids because it uses

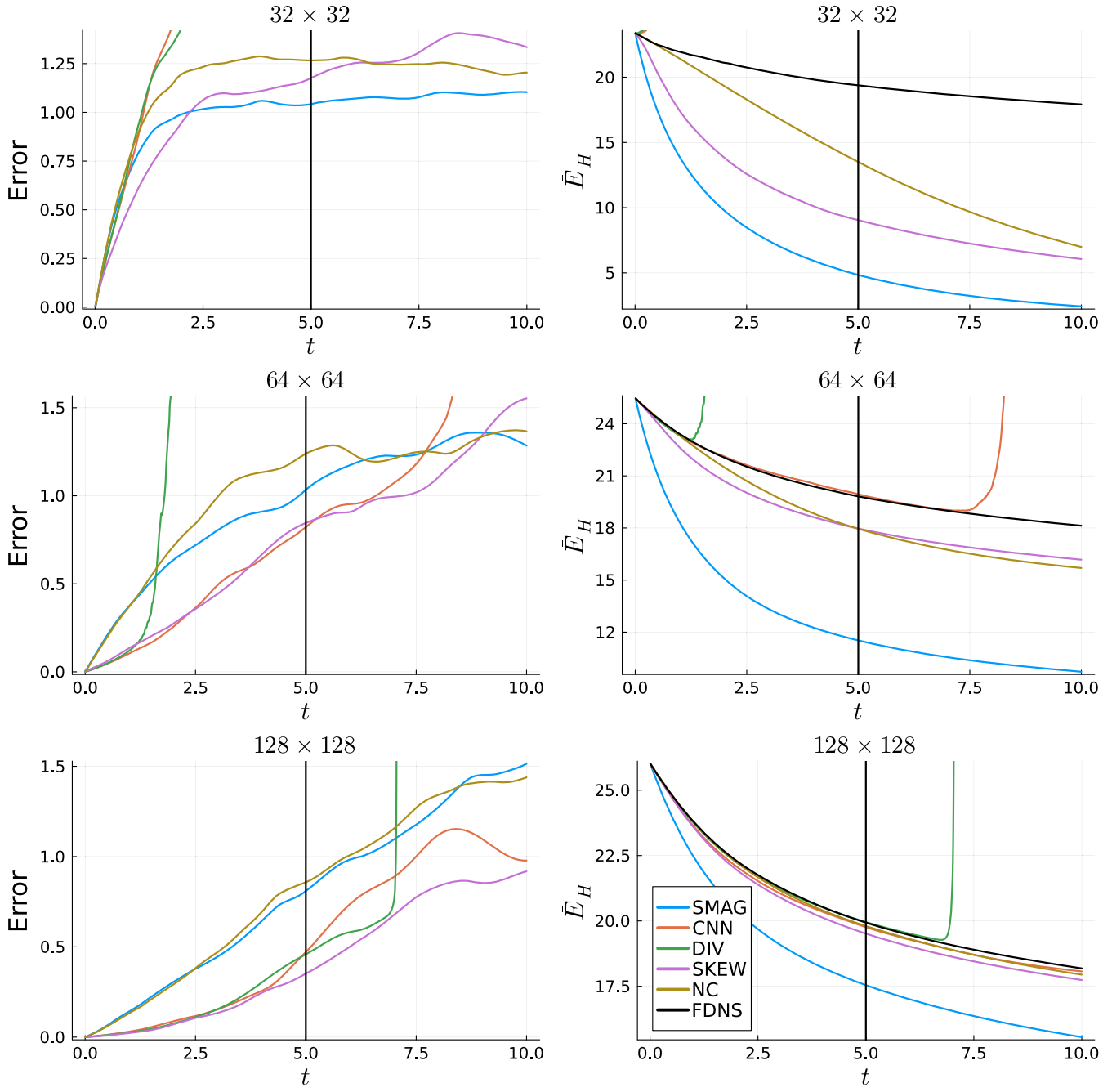

Fig. 4. (Left) Error for each coarse-graining factor as a function of time for the decaying turbulence test case. (Right) Resolved energy trajectories for each coarse-graining factor. For an overview of the methods, see Table 1. The black vertical line indicates $t = 5$. Everything to the right of this line corresponds to extrapolation in time.

only local information. Similarly to the skew-symmetric term, this decomposition could be adjusted by extending the matrix containing the neural network outputs to multiple channels, i.e. $\mathbf{q} = \text{diag}(\mathbf{q}_1, \dots, \mathbf{q}_n)$ A more in-depth motivation behind the proposed neural network architecture is presented in Appendix D, where we relate the choice of $n = d$, where d is the number of spatial dimensions, to a diffusion operator. In this way, the diagonal elements of $\mathbf{q}$ locally control the diffusivity.

4.4. Overview of the closure models

In this section, we introduced a set of four closure models, namely the standard Smagorinsky model (SMAG), a CNN, using a CNN to predict a stress tensor and then taking the divergence (DIV), and finally our skew-symmetric neural network architecture (SKEW). In addition, we also compare to no closure (NC), i.e. $\tilde{\mathbf{c}} = \mathbf{0}_H$. An overview of the closure models and their properties is depicted in Table 1.

5. Results

5.1. Experimental setup

To evaluate the closure models, we consider two test cases: 2D decaying turbulence and Kolmogorov flow. The test cases are inspired by those considered in [15]. For both test cases we consider a periodic domain of $\Omega = [-\pi, \pi] \times [-\pi, \pi]$ and a viscosity of $\nu = \frac{1}{1000}$. Each velocity component of the flow is initialized by a random initial condition only

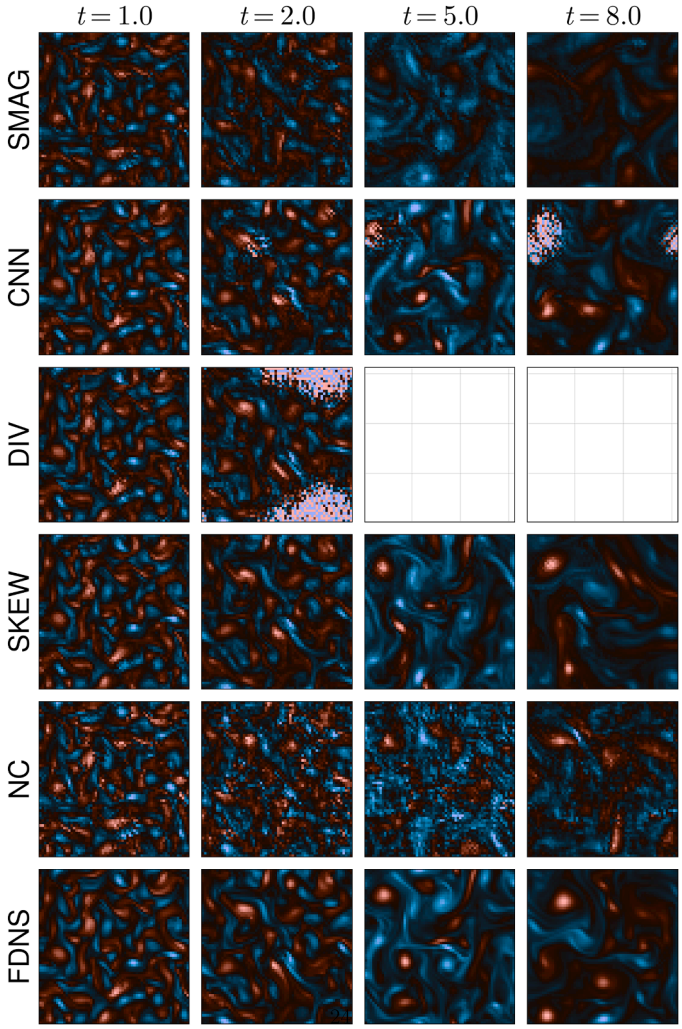

Fig. 5. Vorticity fields at each point in time for each of the closure models on a 64×64 grid. Simulations correspond to the decaying turbulence test case. Blank boxes indicate an unstable simulation.

Table 1
Overview of the different closure models and their properties when combined with the coarse discretization. These closure models consist of the Smagorinsky model (SMAG), a CNN, the divergence of a CNN (DIV), our skew-symmetric neural network architecture (SKEW), and no closure (NC).

	SMAG	CNN	DIV	SKEW (ours)	NC
Mass conservation	✓	✓	✓	✓	✓
Momentum conservation	✓	✗	✓	✓	✓
Dissipative	✓	✗	✗	✓	✓

containing energy in the low wavenumbers (below $\kappa_{\max} = 10$):

$$\mathbf{u}(\mathbf{x}, 0) = \mathrm{Re}\left(\begin{bmatrix} \sum_{\{\boldsymbol{\kappa} \in \mathbb{Z}^2 | 0 < ||\boldsymbol{\kappa}||_2 < \kappa_{\max}\}} c^u_{\boldsymbol{\kappa}} e^{i\boldsymbol{\kappa}\cdot\mathbf{x}} \\ \sum_{\{\boldsymbol{\kappa} \in \mathbb{Z}^2 | 0 < ||\boldsymbol{\kappa}||_2 < \kappa_{\max}\}} c^v_{\boldsymbol{\kappa}} e^{i\boldsymbol{\kappa}\cdot\mathbf{x}} \end{bmatrix} \right), \tag{36}$$

where the real and complex components of the coefficients $c^u_{\boldsymbol{\kappa}}, c^v_{\boldsymbol{\kappa}} \in \mathbb{C}$ are sampled uniformly from the interval $(-1, 1)$. Finally, the coefficients are scaled such that the normalized kinetic energy at $t = 0$, namely

$$\frac{1}{2|\Omega|} \int_\Omega ||\mathbf{u}(\mathbf{x}, 0)||_2^2 \, \mathrm{d}\Omega,$$

equals 1.2. Increasing this initial energy amplifies the characteristic velocity scales of the flow, thereby increasing the effective Reynolds number $\mathrm{Re}_{\mathrm{eff}} = UL/\nu$, where U is a representative velocity magnitude and L a characteristic length scale of the largest eddies. A higher $\mathrm{Re}_{\mathrm{eff}}$ leads to a broader separation of scales and stronger non-linear interactions, which make closure modeling more challenging. The chosen value of 1.2 was found, through preliminary numerical tests, to yield sufficiently rich turbulent dynamics while remaining numerically stable and allowing meaningful comparison between closure models. Before the simulation starts, the sampled initial condition is projected onto a divergence-free basis.

The training data set consists of five simulations starting from an initial condition sampled in this manner. For the DNS we use a computational grid of resolution 2048×2048 and a time step size $\Delta t = 2 \times 10^{-4}$, as in [15]. For the time integration of both the DNS and the closure model-based simulations, we use a 4^{th}-order Runge-Kutta integration scheme [54,55]. For training purposes, we save a snapshot every 10 time steps until $t = 5$. Regarding coarse-graining, we consider three different resolutions, namely 32×32, 64×64, and 128×128, and a time step size of $\overline{\Delta t} = 10\Delta t$, as coarser grids allow for larger time steps [46,56]. For each coarse-graining factor, the machine learning models are trained to reproduce the true solution, i.e., minimize (20), for $n = 5$ time steps. To optimize the closure model parameters, we use the ADAM optimization algorithm [47] with a learning rate of 10^{-3}, decay rates for the first and second momentum estimates at 0.9 and 0.999, respectively, and a mini-batch size of 20. We optimize for in total 500 epochs. These hyperparameter settings resulted in smooth convergence, see E. We consider further hyperparameter studies to be outside the scope of this research, as this research focuses on the neural network architecture. We implemented the closure models in the Julia programming language [57] using the Flux.jl package [58,59]. For each of the CNN-based closure we use four input channels, namely $\bar{\mathbf{u}}_H$ and $\mathbf{m}_H(\bar{\mathbf{u}}_H)$, each containing two channels. The intermediate layers of the CNNs each contain 32 channels, with a total of four intermediate layers (same amount as in [14]). The convolutions in each layer have a radius of $r = 2$. The number of parameters is dominated by the number of hidden layers and channels, which is the same for each of the neural network-based closure models. In this way, the comparison is carried out as fairly as possible. In between the hidden layers, we employ a ReLU activation function and a linear activation function at the final layer [48]. The purely CNN-based closure simply has two output channels to model the closure term in each spatial direction. For DIV we have three output channels, two for the diagonal elements of the stress tensor and one for the off-diagonal ones, due to the symmetry of the true stress tensor [10,17]. For SKEW the underlying CNN has four output channels corresponding to $\mathbf{k}_1, \mathbf{k}_2, \mathbf{q}_1, \mathbf{q}_2$. The convolutions in the $\mathcal{B}$ matrices are chosen to have a convolution radius of $r = 2$. Training of a single neural network takes roughly two hours on an A100 GPU of the Dutch National supercomputer Snellius [60]. Regarding the optimization of the Smagorinsky constant, we choose the constant value that minimizes the L_2-norm of the energy spectra for the training data at $t = 2$ in $\log_{10}$ space. The energy spectra are determined by computing a fast Fourier transform (FFT) of the velocity field. The wavenumbers are then divided into dyadic bins, and the energy belonging to the wavenumbers in the bin is summed to produce the spectrum. This procedure is described in [14,61]. The considered values for the Smagorinsky constant are 0.0 to 0.30 in intervals of 0.01. The obtained optimal values for C_s at each resolution are 0.23, 0.22, and 0.18 for 32×32, 64×64, and 128×128, respectively. These values are within the range of what is typically used for 2D turbulence, namely around 0.18 [62,63].

5.2. Decaying turbulence

The first test case we consider is decaying turbulence, i.e., there is no forcing. Here we consider a single simulation starting from a randomly generated initial condition, generated according to (36). The simulations are carried out up to the final time $t = 10$. Note this is twice as long as present in the training data. Therefore, the second half of the simulation corresponds to extrapolation in time. For each

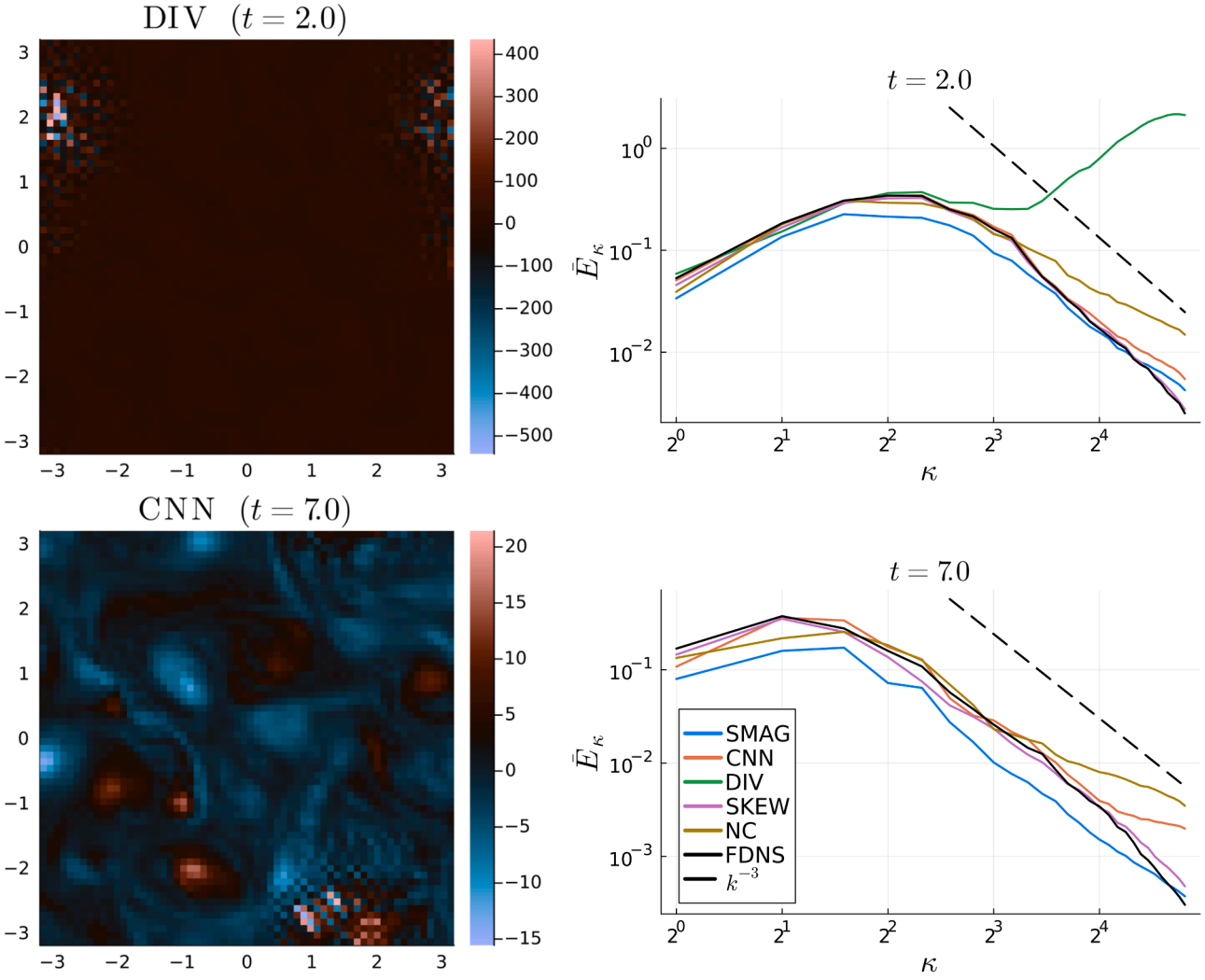

Fig. 6. (Left) Two snapshots where we see numerical oscillations occurring for DIV and CNN. The oscillations eventually result in instabilities. (Right) Energy spectra at the time of these snapshots. The numerical oscillations cause a clear increase in energy in the high wavenumbers.

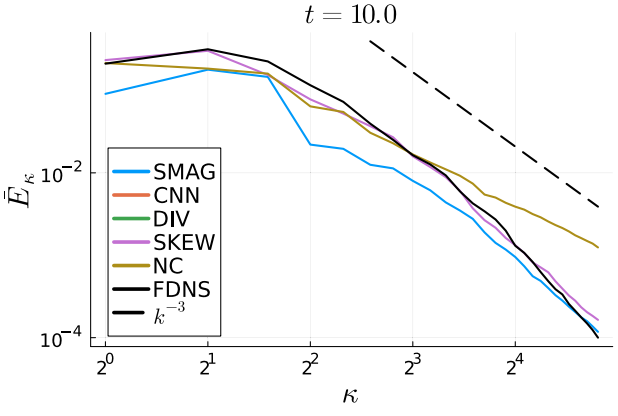

Fig. 7. Energy spectra at the end of the decaying turbulence simulation at $t = 10$. Both the CNN and DIV have become unstable at this point and are therefore omitted from the figure.

coarse-graining factor, the resolved energy and error trajectories are presented in Fig. 4.

The error is defined as

$$\text{error} := \sqrt{\frac{||\bar{\mathbf{u}}_H - \bar{\mathbf{u}}_H^{\text{model}}||_2^2}{||\bar{\mathbf{u}}_H||_2^2}}, \tag{37}$$

where $\bar{\mathbf{u}}_H$ is the FDNS result and $\bar{\mathbf{u}}_H^{\text{model}}$ is the model prediction. We observe that initially the unconstrained machine learning approaches (CNN and DIV) produce the best energy trajectories for resolutions 64×64 and 128×128, whereas SKEW is too dissipative and the Smagorinsky model is even more dissipative. However, as the simulation progresses, the unconstrained machine learning methods often suddenly become unstable. For a resolution of 32×32, these approaches tend to diverge rather quickly, whereas finer resolutions can stay stable over a more extended time period. That said, this seems merely a postponement of the inevitable, as unconstrained machine learning acting on a finer resolution remains highly prone to a sudden and unpredictable catastrophic failure. One can hope for a one-off stable simulation, as shown by the 128×128 resolution CNN. This particular simulation remained stable during the entire considered time period, producing a good energy trajectory and an improved error with respect to NC and SMAG. However, in the vorticity fields presented in Fig. 5, we already notice a build-up of numerical noise for the CNN at the end of the simulation, which may well result in instabilities in the future. In fact, in Section 5.3 we show that this is indeed the expected outcome. As such,

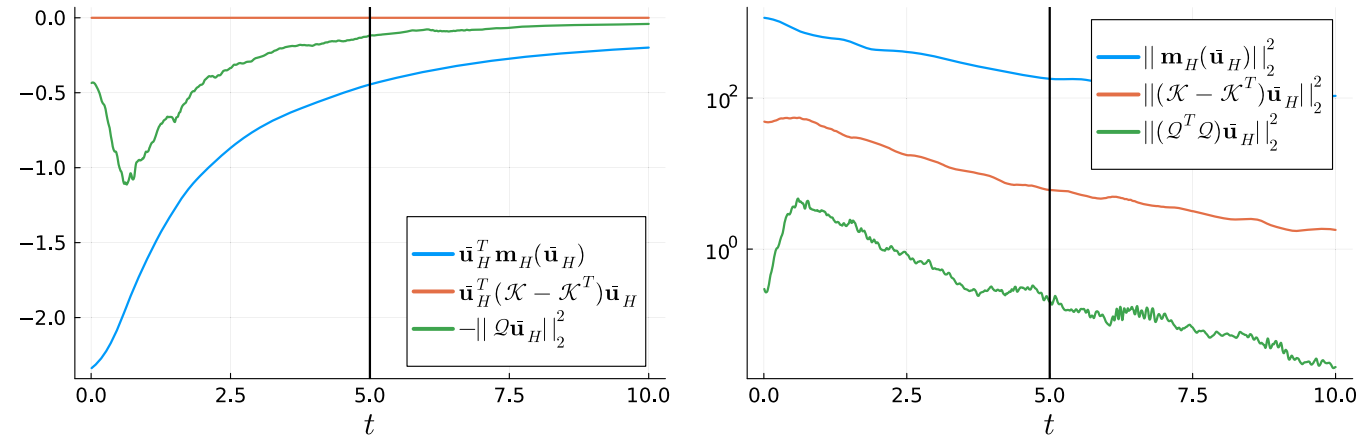

Fig. 8. (Left) Energy contribution for each of the terms in the SKEW architecture along with the contribution of the coarse discretization. (Right) Magnitude of each of the terms. The black vertical line indicates $t = 5$. Everything to the right of this line corresponds to extrapolation in time.

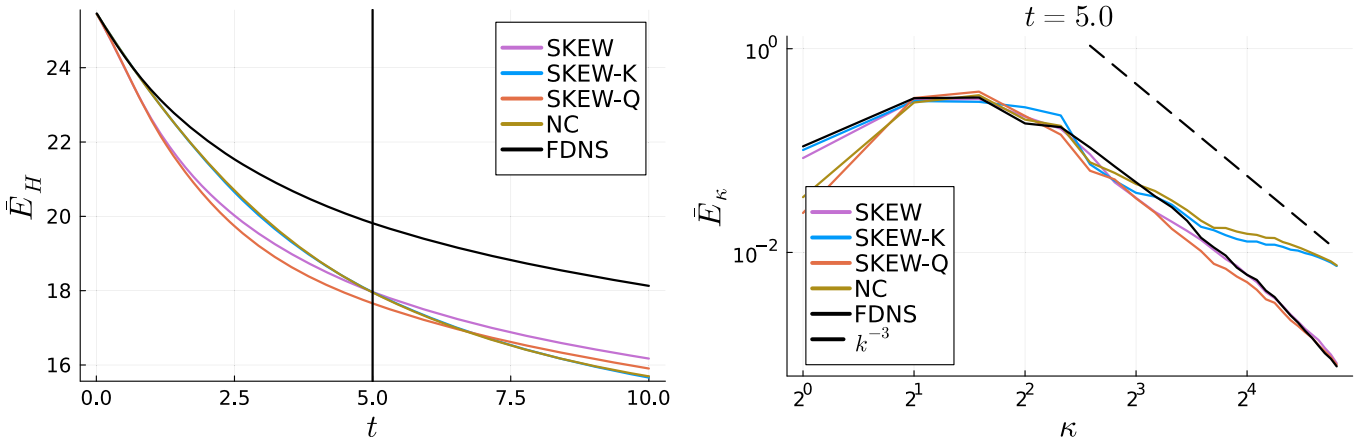

Fig. 9. (Left) Resolved energy trajectories for the full SKEW closure model, SKEW-K (skew-symmetric term only), and SKEW-Q (dissipative term only). The black vertical line indicates $t = 5$. Everything to the right of this line corresponds to extrapolation in time. (Right) Energy spectra halfway through the decaying turbulence simulation at $t = 5$.

the purely data-driven, physics-blind machine learning methods considered here are unsuitable as a viable closure modeling approach for long-term predictions, incapable of outperforming existing physics-based closure models.

On the other hand, for our constrained approach SKEW, we find it consistently provides improved error trajectories with respect to NC, except at a resolution of 32×32. However, it is too dissipative, as stated earlier. This likely comes from the fact that it is constrained to not create energy. This means energy is solely redistributed through the skew-symmetric term and dissipated through the negative-definite term. Especially at large coarse-graining factors, this becomes more problematic, see resolution of 32×32. This is likely caused by the fact that backscatter is more prevalent, see Fig. 3, as more information is being discarded to the subgrid scales. At a resolution of 64×64 SKEW is also too dissipative, however, it does produce an improved error trajectory with respect to the other closures, even in the extrapolation region $t \in (5, 10]$. However, near the end of the simulation, the error becomes larger than for NC and SMAG. Later in this section, we consider the energy spectra to compare the velocity fields in a more statistical fashion, see Figs. 6 and 7.

Carrying out the DNS simulation took roughly 45 min on an A100 GPU, whereas the closure model-based simulations took between 3 and 4 min (with a training time of roughly two hours), for all closure models, see Table 2. This amounts to a computational speed-up of more than 10× with respect to the DNS. For these coarse resolutions, the evaluation time seems to be dominated by computational overhead, as we did not observe a big difference in computation time between the different closure models and resolutions. All closure model-based simulations require more computation time than NC, where SKEW consistently takes the most time. This is likely caused by the additional convolutions in the $\mathcal{B}$ matrices.

Table 2
Computation time in seconds for the different closure models for the decaying turbulence test case. For an overview of the methods, see Table 1.

$\bar{N}$	SMAG	CNN	DIV	SKEW	NC	DNS
32×32	182.69	186.64	189.99	211.39	170.42	2559.88
64×64	210.46	216.26	203.76	225.79	165.27	2559.88
128×128	195.29	200.06	217.75	243.32	187.82	2559.88

Next, we look at the scalar vorticity field $\omega = (\nabla \times \mathbf{u})_3$, at different points in time, produced by the closure models. For a resolution of 64×64, these are depicted in Fig. 5.

The other resolutions are depicted in F. For the remainder of this text, we will stick to a resolution of 64×64, as we believe this is a nice middle ground between the poor results obtained for 32×32 and almost perfect

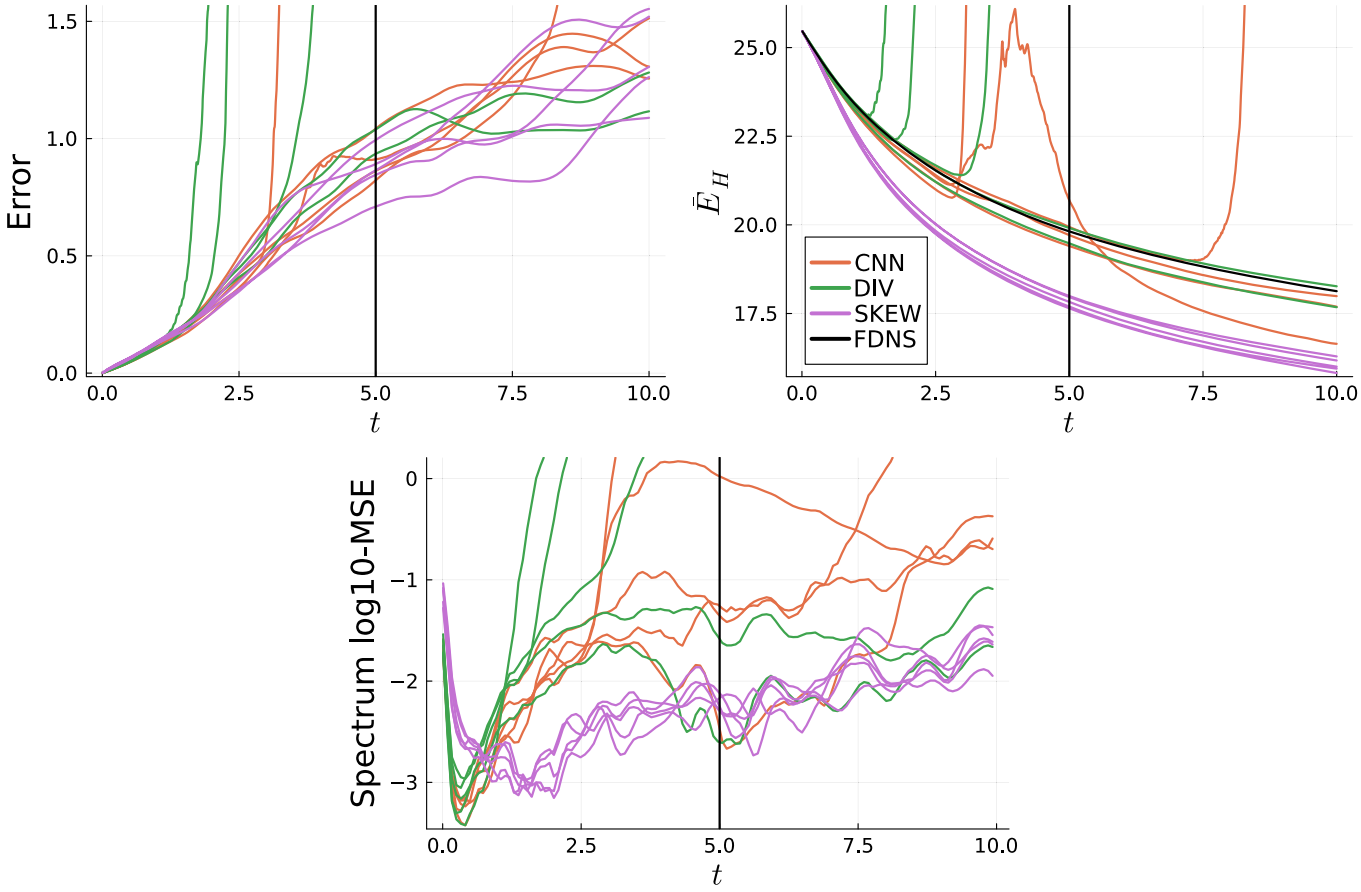

Fig. 10. (Top-left) Error over time for each closure model in the ensemble of five. (Top-right) Resolved energy trajectories for each closure model in the ensemble of five. (Bottom) Error of the energy spectrum over time calculated by computing the spectrum in $\log_{10}$ space, then computing the MSE with respect to the FDNS spectrum, and finally reporting the $\log_{10}$ of this value. These trajectories are also depicted for each closure model in the ensemble of five. The black vertical line indicates $t = 5$. Everything to the right of this line corresponds to extrapolation in time.

reproduction for 128×128. This coincides with the coarse-graining factor applied in [15]. Regarding the vorticity fields, we find that initially all three machine learning closures nicely match the FDNS result. However, as the simulation progresses, we observe a build-up of numerical noise for the unconstrained machine learning closures. This eventually leads to unstable simulations. For SKEW this does not happen. Even after diverging from the FDNS, it still produces smooth results which seem to match the FDNS on a qualitative level. The instabilities in the unconstrained machine learning approaches can possibly be alleviated by adding more training data or adding noise to the training [36,38]. However, we argue that given the same amount of training data, using SKEW has clear stability benefits.

To assess the physical consistency of the produced trajectories, we compare the energy spectra at different points during the simulation. This is done, as a pointwise error, such as presented in Fig. 4, no longer provides relevant information after the solutions diverge from the FDNS [15]. The spectra are depicted in Fig. 6.

Here we observe that SMAG is too dissipative at the large scales, but nicely reproduces the κ^{-3} decay of the energy spectrum expected from 2D turbulence [64]. Regarding SKEW we find it has a better overall fit with the FDNS, as compared to SMAG, for both the high and low wavenumbers. Regarding the unconstrained machine learning approaches, we observe a clear buildup in energy in the large wavenumbers. From the depicted vorticity fields, we can clearly see the numerical noise responsible for this.

In Fig. 4, we observed that SKEW reaches a larger error than NC and SMAG at the end of the simulations. To assess whether or not the simulation produced by SKEW is still physically consistent, we consider the energy spectrum at this point. This is depicted in Fig. 7.

Here we observe that even though the error, see (37), is larger, SKEW still produces an energy spectrum that more closely matches the FDNS spectrum, as compared to SMAG and NC. It also produces the expected κ^{-3} slope. SMAG also achieves this, but underestimates the energy in all wavenumbers, whereas NC suffers from a build-up of energy in the large wavenumbers. The latter likely corresponds to numerical noise, due to the coarseness of the grid. After a while SKEW diverges from the FDNS. This behavior is expected given the characteristic sensitivity on initial conditions of the two-dimensional Navier-Stokes equations in the turbulent regime [65]. Nevertheless, the SKEW model continues to produce physically consistent statistics and spectra, even when extrapolating in time.

Finally, we consider the contributions of both the skew-symmetric and negative-definite term in the SKEW architecture. We consider both the energy contribution and the magnitude of the terms. This is depicted in Fig. 8.

The first observation is that the energy contribution of the skew-symmetric term is indeed zero, as it should be. Furthermore, we find that the negative-definite term is slightly less dissipative than the coarse discretization. The trajectory also has a different shape; the dissipation coming from the negative-definite term peaks around $t = 1$, whereas the dissipation from the coarse discretization decreases smoothly over time. The shape of the dissipation trajectory of the negative-definite term might arise from the transition of the flow from one physical regime to another. More specifically, the initial condition contains energy only

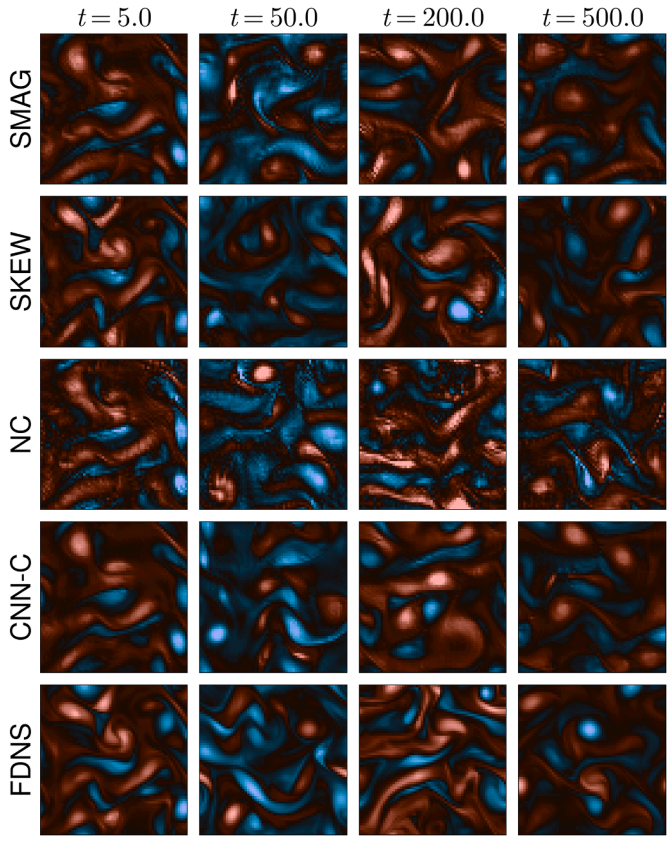

Fig. 11. Vorticity fields at each point in time for each of the closure models on a 64×64 grid. Simulations correspond to the Kolmogorov flow test case.

at low wavenumbers. This means the flow undergoes a transient period before the formation of its characteristic slope, see Fig. 6. Next, we look at the magnitude of the different terms. Here we find that the skew-symmetric term has a larger magnitude than the negative-definite term. This means it has a more significant impact on the simulation. This supports the use of a skew-symmetric term in the closure model.

To conclude this section, we also examine the effect of omitting either the skew-symmetric term or the negative-definite term from the SKEW architecture. To this end, we perform the decaying turbulence simulation with one of the two terms set to zero. The resulting resolved energy trajectories and energy spectra are shown in Fig. 9.

We observe that omitting the negative-definite term leads to markedly less dissipative behavior, as expected. The resulting energy trajectory is nearly identical to that of NC. The corresponding energy spectrum matches the FDNS result well at low wavenumbers; however, at high wavenumbers, a clear build-up of energy appears, resembling the NC spectrum and indicating numerical noise. Conversely, omitting the skew-symmetric term produces more dissipative behavior and yields physically consistent energy levels at high wavenumbers, with the characteristic κ^{-3} slope accurately recovered. Nonetheless, this variant over-dissipates energy at the low wavenumbers.

From these simulations, we conclude that the skew-symmetric term is essential for accurately capturing the energy at low wavenumbers, whereas the negative-definite term plays a crucial role in suppressing numerical noise and recovering the characteristic κ^{-3} slope. This means both terms make significant and complementary contributions to the accuracy and physical consistency of the simulation.

5.3. Consistency of closure model performance

Training neural networks is inherently random, due to the selection of mini-batches, initialization of the weights, etc. This is why we instantiate multiple replicas of each network to fully assess their potential as a closure model. For this purpose, we train an ensemble of five replicas for each neural network architecture. This allows us to evaluate the consistency of the training procedure in terms of producing good closure models. Before evaluating the networks, we ensured no convergence issues occurred during training. The resulting error and energy trajectories, for each neural network, evaluated on the decaying turbulence test case, are depicted in Fig. 10.

Regarding the plain CNN architecture, we observe that two out of five networks result in unstable simulations, and for DIV, three out of five. Hence, it is worth pointing out that, therefore, simply retraining exactly the same machine-learning architecture can be the difference between a stable simulation and a numerical failure, highlighting the fickle nature of these methods. For SKEW, no ensemble members became unstable.

Regarding the error trajectories, we observe similar performances for all the networks that remained stable, whereas the energy trajectories were best reproduced by some of the unconstrained machine learning closures. However, to evaluate the build-up of numerical noise and physical consistency, we computed the error in energy spectrum during the simulation. This is also depicted in Fig. 10. Here we find that the SKEW architecture consistently outperforms the other architectures. From this, we conclude that our SKEW neural network architecture is not only guaranteed to be stable, but also consistently produces physical results, without a build-up of numerical noise.

5.4. Kolmogorov flow

Next, we aim to evaluate the long-term performance of the closure models and their extrapolation capabilities. To do this, we require a different test case, as decaying turbulence from the previous test case eventually decays to zero. We therefore consider Kolmogorov flow, with the same viscosity $\nu = \frac{1}{1000}$ and periodic domain $\Omega = [-\pi, \pi] \times [-\pi, \pi]$. Kolmogorov flow is characterized by the following forcing:

$$\mathbf{f}(\mathbf{u}, \mathbf{x}, t) = \begin{bmatrix} \sin(4y) \\ 0 \end{bmatrix} - 0.1\mathbf{u}, \tag{38}$$

and is often used to evaluate machine learning closure models [14,15,33]. The machine learning models are not retrained for this test case. This means the models will have to extrapolate from the decaying turbulence training data to a test case that includes forcing. To initialize the simulation, we first carry out a DNS on a 2048×2048 grid until $t = 25$, starting from initial condition (36). This serves as a warm-up of the system, such that it reaches a statistical equilibrium. The final velocity field then serves as an initial condition to evaluate the closure models. We then simulate for 500 model time units starting from this initial condition. This is a significant extrapolation, as the training data was for the interval $t \in [0, 5]$ and for a test case without forcing. In addition to the previously introduced closure models, we apply backscatter clipping to the trained CNN to obtain a stable closure model by removing backscatter. This is done by projecting the CNN output on an eddy-viscosity model. From this, we obtain an eddy-viscosity value $\nu_t^{\text{clipping}}(\mathbf{x})$ such that the CNN output is matched as close as possible in the L_2-norm. Negative values for $\nu_t^{\text{clipping}}(\mathbf{x})$ are then set to zero to provide stability. The clipping procedure is described in [37]. We will refer to this closure model with the acronym CNN-C. Vorticity snapshots from the simulations are depicted in Fig. 11.

Snapshots for CNN and DIV are not depicted, as these simulations become unstable quickly after their initialization, see Fig. 12. For NC we observe a lot of numerical noise, whereas in SMAG (to a lesser extent), SKEW, and CNN-C this is smoothed out. Regarding the snapshot at $t = 5$, we find the filtered DNS results are most closely matched by SKEW.

To make a more thorough comparison, we consider the resolved energy trajectories, along with an average energy spectrum for the simulation, see Fig. 12.

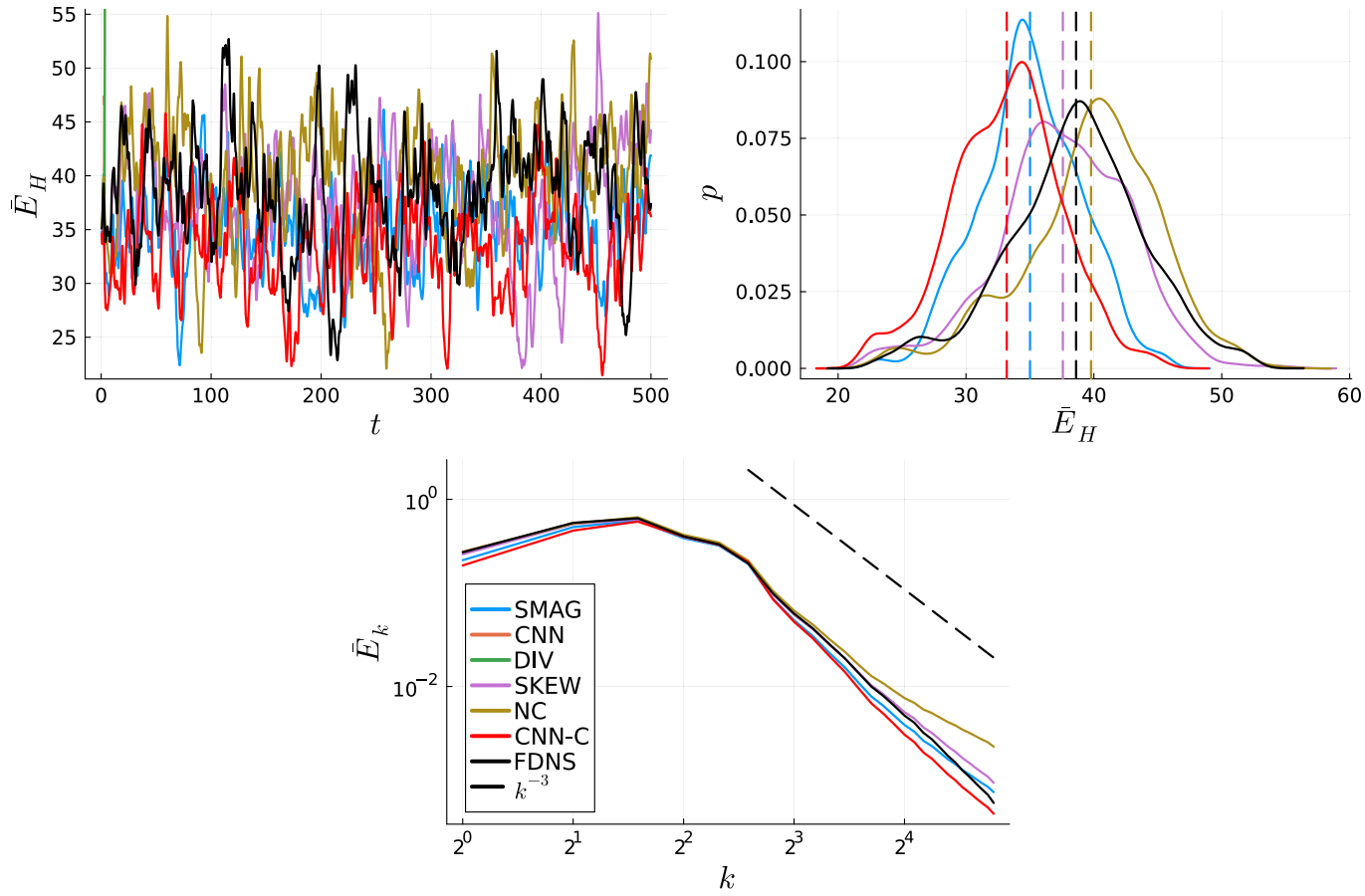

Fig. 12. (Top-left) Resolved energy trajectories for each of the closure models in the Kolmogorov flow test case. (Top-left) Corresponding distributions of the resolved energy for the entire simulation. Dashed vertical lines correspond to the mean. (Bottom) Energy spectra were obtained by first computing the energy spectrum for each snapshot and then computing the average value for each wavenumber. Both the CNN and DIV closure models resulted in unstable simulations, so their spectrum is omitted from the figures.

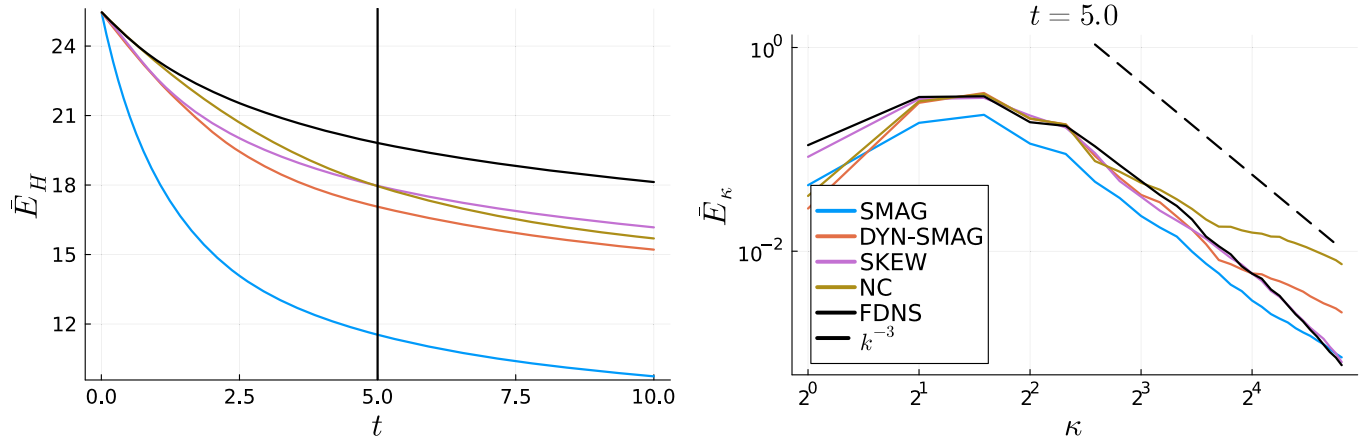

Fig. 13. (Left) Resolved energy trajectories for the decaying turbulence test case, where DYN-SMAG corresponds to the dynamic Smagorinsky model. The black vertical line indicates $t = 5$. Everything to the right of this line corresponds to extrapolation in time. (Right) Energy spectra halfway through the decaying turbulence simulation at $t = 5$.

The first thing we observe from the resolved energy trajectories is that the unconstrained machine learning approaches both become unstable at the start of the simulation. The remaining closure models remain stable, as they are strictly dissipative. To make a statistical comparison of the resulting flow fields, we consider the distribution of the resolved energy for the entire simulation and the average energy spectra. Here we find that SMAG and CNN-C are too dissipative, as the distributions are shifted to the left with respect to FDNS. For CNN-C, this phenomenon is reported in [36]. Both NC and SKEW seem to give a good prediction of both the mean and shape of the distribution.

Looking at the energy spectra, we find that NC indeed produces numerical noise, looking at the energy in the large wavenumbers. In

addition, we find that SKEW performs the best in the low and intermediate wavenumbers, while SMAG performs the best in the high wavenumbers. Overall, we find that for this extrapolation test case SKEW performs, at worst, as well as SMAG. From this, we conclude that SKEW is both stable and accurate, and is capable of extrapolating to different test cases, without being retrained.

5.5. comparison to the dynamic smagorinsky model

To provide a comparison between our SKEW architecture and a more robust physics-based closure model, we consider the dynamic Smagorinsky model. The formulation and underlying theory of this model are described in [22]. The main idea is that the Smagorinsky constant in (22) is determined dynamically and depends on space, time, and the resolved velocity field, i.e. $C_s = C_s(\mathbf{x}, t, \bar{\mathbf{u}})$.

The dynamic procedure builds on the assumption that the subgrid-scale stresses are self-similar across filter scales and that the same eddy-viscosity ansatz holds at both the grid filter and a larger test filter. By enforcing consistency between these two levels, the model computes C_s from the resolved flow itself rather than prescribing it a priori. This allows the model to reduce dissipation in laminar or well-resolved regions while increasing it where subgrid activity is strong. Its main strength is therefore that it provides dissipation only where it is needed, making it significantly less dissipative than the standard Smagorinsky model.

The resulting energy trajectory and energy spectrum for the dynamic Smagorinsky model in the decaying turbulence test case are shown in Fig. 13, together with the results from our SKEW architecture.

For the energy trajectory, we find that the dynamic Smagorinsky model is indeed less dissipative than the standard Smagorinsky model (SMAG). However, it remains slightly more dissipative than our SKEW architecture, which matches the FDNS result the closest. Regarding the energy spectrum, the dynamic Smagorinsky model provides a substantial improvement over SMAG and NC. Nonetheless, it still underpredicts energy at low wavenumbers and overpredicts energy at high wavenumbers, whereas our SKEW architecture does not exhibit these deficiencies.

5.6. Limitations of machine learning closure models

Note that a neural network is not strictly necessary to enforce a skew-symmetric framework. Traditional calibration or data assimilation approaches could be used, which typically assume a fixed functional form and tune a limited set of coefficients to DNS data. In contrast, a neural network provides a flexible functional representation capable of capturing complex, non-linear dependencies of the closure term on local flow features, dependencies that are difficult to express or calibrate explicitly using fixed parametric forms. While the skew-symmetric structure enforces energy conservation, the entries of the skew-symmetric matrix (as well as those of the dissipative term) are learned functions of the flow state, allowing the model to adapt dynamically to local flow conditions rather than relying on globally calibrated coefficients.

Finally, it should be noted that fundamental challenges exist when employing machine learning within the context of LES (or any physics-based simulation) [66]. Neural networks easily have millions (or more) tunable parameters with no physical meaning, are data-hungry, and act as black boxes, making them more complex than their physics-based counterparts. They are also prone to instabilities over long integration times, an issue that we have explicitly addressed in the present manuscript. Our proposed framework, therefore, represents a step towards embedding physical constraints (in this case, guaranteed energy conservation) directly into the architecture of a data-driven model. This combination of physical structure and data-driven flexibility embodies a key advantage of physics-informed machine learning: it ensures physical consistency while retaining the expressive power of modern machine-learning methods. That said, many of the aforementioned issues remain and require further study.

6. Conclusion

In this work, we started off by exploring the conservation laws inherent in the incompressible Navier-Stokes equations, specifically, mass, momentum, and energy conservation. We employed a discretization that preserves these laws in a discrete sense. To coarse-grain the simulation, we used a face-averaging filter, ensuring that the resulting coarse-grained velocity field continues to satisfy mass conservation. We then examined different approaches to modeling the commutator error introduced by coarse-graining. We first considered the Smagorinsky model, which is strictly dissipative, followed by more advanced machine learning approaches based on convolutional neural networks, capable of modeling backscatter. However, these unconstrained models lack stability guarantees. To address this limitation, we introduced our skew-symmetric neural architecture [13]. This architecture enforces stability while increasing model freedom from the negative-definite eddy-viscosity basis by introducing a skew-symmetric term. A change from our previous work [13] is that the subgrid-scale energy is no longer explicitly modeled. In addition, we tackled much more challenging 2D turbulence applications, as opposed to simple 1D test cases considered in [13]. Based on offline analysis on the training data, we hypothesized that dissipative closure models, such as the skew-symmetric architecture we introduce, are likely to perform well for the considered test cases.

We evaluated our closure models across three coarse-graining factors, from a 2048×2048 grid down to resolutions of 128×128, 64×64, and 32×32. The closure models were tested in a decaying turbulence simulation with an initial condition different from those in the training data and over an extended simulation time. We found that none of the models performed well at the largest coarse-graining factor (from 2048×2048 down to 32×32). Initially, the unconstrained machine learning models provided promising results, even outperforming our skew-symmetric model in kinetic energy predictions. However, numerical errors accumulated, leading to instability. Trajectory fitting, i.e., training the closure models to reproduce the filtered DNS solution, alone is therefore not enough to guarantee stable closure models. In contrast, our skew-symmetric model remained stable throughout, albeit at the cost of a larger dissipation rate. Despite the increased dissipation, we argue that stability is a worthwhile trade-off. In addition, our skew-symmetric architecture outperformed the Smagorinsky model in this test case and reproduced the expected k^{-3} in the energy spectrum for 2D turbulence. Furthermore, our architecture also outperformed the dynamic Smagorinsky model, a widely used and well-established physics-based closure model, in this test case.

To account for the inherent randomness in training neural networks, we trained five instances of each model with different weight initializations and mini-batch selections. All instances of the skew-symmetric model remained stable and accurate, while conventional machine learning models exhibited significant instabilities. This highlights the improved consistency of our approach in training robust closure models.

We further assessed the models on the Kolmogorov flow test case, which was not represented in the training data. The unconstrained machine learning models again suffered from instabilities, whereas our skew-symmetric closure model successfully captured the correct energy spectrum when averaged over time. In this regard, it performed comparably to the Smagorinsky model, which proved to be well-suited to this test. We also applied backscatter clipping to the trained CNN. This resulted in a stable simulation; however, it did not perform better than the skew-symmetric architecture or the Smagorinsky model.

Overall, our results demonstrate that the skew-symmetric architecture we introduced here significantly enhances the stability of machine-learning-based closure models, albeit at the cost of increased dissipation. We believe this work represents a step toward enabling long-time simulations with machine learning closures.

For future work, several directions merit exploration. The treatment of boundary conditions, particularly the padding used for convolutional neural networks, requires careful attention. Extending our approach to

unstructured grids is another promising avenue, where graph neural networks could provide a useful framework [67]. Finally, introducing an additional energy source to counteract the absence of backscatter from the skew-symmetric architecture could be beneficial, though careful clipping mechanisms would be needed to prevent instabilities. Explicitly modeling the subgrid-scale energy, as in [13], is another logical extension.

Declaration of generative AI and AI-assisted technologies in the writing process

During the preparation of this work the author(s) used ChatGPT in order to improve language and grammar. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

CRediT authorship contribution statement

T. van Gastelen: Writing – original draft, Software, Methodology, Conceptualization; **W. Edeling:** Writing – review & editing; **B. Sanderse:** Writing – review & editing, Methodology, Funding acquisition, Conceptualization.

Data availability

The code used to generate the training data and the implementation of the neural networks will be made available on request.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This publication is part of the project “Unraveling Neural Networks with Structure-Preserving Computing” (with project number OCENW.GROOT.2019.044 of the research programme NWO XL which is financed by the Dutch Research Council (NWO)). Part of this publication is funded by Eindhoven University of Technology. Finally, we thank the reviewers for their feedback, enhancing the quality of the article.

Acronyms

BC boundary condition.

CNN convolutional neural network.

CNN-C convolutional neural network with backscatter clipping.

DIV divergence of stress tensor predicted by a neural network.

DNS direct numerical simulation.

FDNS filtered direct numerical simulation.

FFT fast Fourier transform.

LES large eddy simulation.

MSE mean squared error.

NC no closure.

PDE partial differential equation.

RHS right-hand side.

SKEW skew-symmetric neural network architecture.

SMAG Smagorinsky model.

Appendix A. Physical structure of the Navier-Stokes equations

The Navier-Stokes equations, see (1a), represent a set of fundamental physical laws, namely: conservation of mass, momentum, and energy (for zero dissipation). These are collectively referred to as the physical structure of the system. Conservation of mass can easily be shown by computing the change of mass in a volume $\mathcal{V}$ due to the flux across the surface S. This reads

$$\oint_S \mathbf{u}\cdot\mathbf{n}\mathrm{d}S = \int_{\mathcal{V}} \nabla\cdot\mathbf{u}\mathrm{d}V = 0, \tag{A.1}$$

where we used the divergence theorem to write the surface integral as a volume integral. $\mathbf{n}$ represents the outward unit normal vector of S and dS denotes the corresponding scalar surface-area element.

Conservation of momentum is also straightforward to derive: The momentum is defined as

$$\mathbf{P} = \int_\Omega \mathbf{u}\mathrm{d}\Omega, \tag{A.2}$$

where Ω is the spatial domain. The change in momentum is computed as follows:

$$\begin{aligned}\frac{\mathrm{d}\mathbf{P}}{\mathrm{d}t} &= \int_\Omega \frac{\partial\mathbf{u}}{\partial t}\,\mathrm{d}\Omega \\ &= -\oint_{\partial\Omega}(\mathbf{u}\mathbf{u}^T)\cdot\mathbf{n}\mathrm{d}S - \oint_{\partial\Omega} p\mathbf{n}\,\mathrm{d}S + \nu\oint_{\partial\Omega}\nabla\mathbf{u}\cdot\mathbf{n}\mathrm{d}S + \int_\Omega \mathbf{f}\,\mathrm{d}\Omega \\ &= \int_\Omega \mathbf{f}\,\mathrm{d}\Omega,\end{aligned} \tag{A.3}$$

where we filled in (1a) and rewrote most of the terms to boundary integrals. These disappear on periodic domains. This means that the momentum in each direction only changes due to the body force.

Finally, the total (kinetic) energy in the system is defined as

$$E = \frac{1}{2}\int_\Omega \mathbf{u}\cdot\mathbf{u}\mathrm{d}\Omega. \tag{A.4}$$

Using the product rule, we can write the change in energy as

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \int_\Omega \mathbf{u}\cdot\frac{\partial\mathbf{u}}{\partial t}\mathrm{d}\Omega = \int_\Omega \mathbf{u}\cdot(-\nabla\cdot(\mathbf{u}\mathbf{u}^T) - \nabla p + \nu\nabla^2\mathbf{u} + \mathbf{f})\mathrm{d}\Omega. \tag{A.5}$$

The pressure and friction contributions can be simplified using integration by parts:

$$\int_\Omega -\mathbf{u}\cdot\nabla p\mathrm{d}\Omega = \int_\Omega p(\nabla\cdot\mathbf{u})\mathrm{d}\Omega = 0, \tag{A.6}$$

$$\int_\Omega \nu\mathbf{u}\cdot\nabla^2\mathbf{u}\mathrm{d}\Omega = -\int_\Omega \nu||\nabla\mathbf{u}||_2^2\mathrm{d}\Omega, \tag{A.7}$$

where we used the fact $\nabla\cdot\mathbf{u} = 0$ and that the boundary contributions cancel on periodic domains.

To find the energy contribution for the convective term, we start off by rewriting it using the product rule for outer products:

$$\nabla\cdot(\mathbf{u}\mathbf{u}^T) = (\mathbf{u}\cdot\nabla)\mathbf{u} + (\nabla\cdot\mathbf{u})\mathbf{u} = (\mathbf{u}\cdot\nabla)\mathbf{u}, \tag{A.8}$$

where the second term vanishes due to divergence freeness. Next, we rewrite the term using a vector calculus identity:

$$(\mathbf{u}\cdot\nabla)\mathbf{u} = \frac{1}{2}\nabla(\mathbf{u}\cdot\mathbf{u}) - \mathbf{u}\times(\nabla\times\mathbf{u}). \tag{A.9}$$

We then take the inner product with $\mathbf{u}$ to obtain

$$\mathbf{u}\cdot\nabla\cdot(\mathbf{u}\mathbf{u}^T) = \mathbf{u}\cdot\frac{1}{2}\nabla(\mathbf{u}\cdot\mathbf{u}) - \mathbf{u}\cdot(\mathbf{u}\times(\nabla\times\mathbf{u})) = \mathbf{u}\cdot\frac{1}{2}\nabla(\mathbf{u}\cdot\mathbf{u}), \tag{A.10}$$

where the second term cancels due to the fact that the cross product between two vectors is orthogonal to both of these vectors. Using the product rule starting from $\nabla \cdot (\mathbf{u}(\mathbf{u} \cdot \mathbf{u}))$ this can be rewritten to

$$\mathbf{u} \cdot \frac{1}{2}\nabla(\mathbf{u} \cdot \mathbf{u}) = \frac{1}{2}\nabla \cdot (\mathbf{u}(\mathbf{u} \cdot \mathbf{u})) - \frac{1}{2}(\mathbf{u} \cdot \mathbf{u})(\nabla \cdot \mathbf{u}) = \frac{1}{2}\nabla \cdot (\mathbf{u}(\mathbf{u} \cdot \mathbf{u})), \tag{A.11}$$

which is in divergence form. Once again, we used the fact that $\mathbf{u}$ is divergence-free to simplify this expression. This term integrates to zero:

$$\int_\Omega \mathbf{u} \cdot \nabla \cdot (\mathbf{u}\mathbf{u}^T) \mathrm{d}\Omega = \int_\Omega \frac{1}{2}\nabla \cdot (\mathbf{u}(\mathbf{u} \cdot \mathbf{u})) \mathrm{d}\Omega = \oint_{\partial\Omega} \frac{1}{2}(\mathbf{u}(\mathbf{u} \cdot \mathbf{u})) \cdot \mathbf{n} \mathrm{d}S = 0, \tag{A.12}$$

due to the divergence theorem and the fact that Ω is a periodic domain. The change in energy is finally written as

$$\frac{\mathrm{d}E}{\mathrm{d}t} = -\int_\Omega \nu ||\nabla \mathbf{u}||_2^2 \mathrm{d}\Omega + \int_\Omega \mathbf{u} \cdot \mathbf{f} \mathrm{d}\Omega, \tag{A.13}$$

which means the energy is always decreasing in the absence of forcing.

Appendix B. Stucture-preserving finite volume discretization

For the employed finite volume discretization, presented in [6], the physical structure of the Navier-Stokes equations is preserved in a discrete sense. Discretely, the total momentum and energy are approximated as

$$\mathbf{P}_h = \mathbb{1}_h \mathbf{\Omega}_h \mathbf{u}_h, \tag{B.1}$$

$$E_h = \frac{1}{2}\mathbf{u}_h^T \mathbf{\Omega}_h \mathbf{u}_h. \tag{B.2}$$

The change in momentum for this discretization is given by

$$\begin{aligned} \frac{\mathrm{d}\mathbf{P}_h}{\mathrm{d}t} &= \mathbb{1}_h \mathbf{\Omega}_h \frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} \\ &= \mathbb{1}_h(-\mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h - \mathbf{G}_h\mathbf{p}_h + \nu\mathbf{D}_h\mathbf{u}_h + \mathbf{\Omega}_h\mathbf{f}_h) \\ &= \mathbb{1}_h \mathbf{\Omega}_h \mathbf{f}_h, \end{aligned} \tag{B.3}$$

as the discrete operators are carefully constructed such that the column vectors sum up to zero. This means momentum conservation is satisfied by this discretization. Using the product rule, we obtain the change in energy as

$$\begin{aligned} \frac{\mathrm{d}E_h}{\mathrm{d}t} &= \mathbf{u}_h^T \mathbf{\Omega}_h \frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} \\ &= \mathbf{u}_h^T(-\mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h - \mathbf{G}_h\mathbf{p}_h + \nu\mathbf{D}_h\mathbf{u}_h + \mathbf{\Omega}_h\mathbf{f}_h) \\ &= -\mathbf{u}_h^T\mathbf{Q}_h^T\mathbf{Q}_h\mathbf{u}_h + \mathbf{u}_h^T\mathbf{\Omega}_h\mathbf{f}_h = -||\mathbf{Q}_h\mathbf{u}_h||_2^2 + \mathbf{u}_h^T\mathbf{\Omega}_h\mathbf{f}_h, \end{aligned} \tag{B.4}$$

where we used the fact that the diffusion operator $\mathbf{D}_h$ can be Cholesky decomposed as $-\mathbf{Q}_h^T\mathbf{Q}_h$ [3,45]. The convective contribution disappears due to the skew-symmetry of the discrete operator:

$$\mathbf{C}_h(\mathbf{u}_h) = -\mathbf{C}_h^T(\mathbf{u}_h) \quad \rightarrow \quad \mathbf{u}_h^T\mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h = -\mathbf{u}_h^T\mathbf{C}_h^T(\mathbf{u}_h)\mathbf{u}_h = 0, \tag{B.5}$$

where we used the symmetry of the inner product. Note that the skew-symmetry of the convection operator is only true for a divergence-free $\mathbf{u}_h$ [14]. The pressure term disappears because the discretization satisfies $\mathbf{G}_h = -\mathbf{M}_h^T$ [45]. Writing the energy contribution we obtain:

$$-\mathbf{u}_h^T\mathbf{G}_h\mathbf{p}_h = \mathbf{u}_h^T\mathbf{M}_h^T\mathbf{p}_h = \mathbf{p}_h^T\mathbf{M}_h\mathbf{u}_h = 0, \tag{B.6}$$

where we used the divergence-free constraint on the velocity field.

Appendix C. Pressure projection

The divergence freeness can also be written as a projection of the PDE discretization (5) on a divergence-free basis [14]. To see this, we compute the divergence of the non-pressure related terms in (5) as

$$\mathbf{M}_h\mathbf{\Omega}_h^{-1}(-\mathbf{C}_h(\mathbf{u}_h)\mathbf{u}_h + \nu\mathbf{D}_h\mathbf{u}_h + \mathbf{\Omega}_h\mathbf{f}_h) = \mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{m}_h(\mathbf{u}_h), \tag{C.1}$$

where we grouped the different terms into $\mathbf{m}_h(\mathbf{u}_h)$. The purpose of the gradient of the pressure is to remove this divergence from the RHS of (5). To obtain the pressure that achieves this, we solve the following linear system:

$$\begin{aligned} &\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{m}_h(\mathbf{u}_h) - \mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{G}_h\mathbf{p}_h = \mathbf{0} \quad \rightarrow \\ &\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{m}_h(\mathbf{u}_h) = \mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{G}_h\mathbf{p}_h \quad \rightarrow \\ &\mathbf{p}_h = (\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{G}_h)^{-1}\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{m}_h(\mathbf{u}_h). \end{aligned} \tag{C.2}$$

By filling this in to (5) we obtain

$$\begin{aligned} &\mathbf{\Omega}_h\frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} = \mathbf{m}_h(\mathbf{u}_h) - \mathbf{G}_h(\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{G}_h)^{-1}\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{m}_h(\mathbf{u}_h) \quad \rightarrow \\ &\mathbf{\Omega}_h\frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} = \underbrace{(\mathbf{I} - \mathbf{G}_h(\mathbf{M}_h\mathbf{\Omega}_h^{-1}\mathbf{G}_h)^{-1}\mathbf{M}_h\mathbf{\Omega}_h^{-1})}_{:=\mathcal{P}_h}\mathbf{m}_h(\mathbf{u}_h) \quad \rightarrow \\ &\mathbf{\Omega}_h\frac{\mathrm{d}\mathbf{u}_h}{\mathrm{d}t} = \mathcal{P}_h\mathbf{m}_h(\mathbf{u}_h), \end{aligned} \tag{C.3}$$

where $\mathcal{P}_h \in \mathbb{R}^{2N\times 2N}$ projects $\mathbf{m}_h(\mathbf{u}_h)$ onto a divergence free basis. This transforms the discretized PDE into a single equation. Note that in practice, we typically solve for the pressure rather than performing the projection in this manner. However, this formulation is more convenient for closure modeling.

Appendix D. Motivation behind skew-symmetric architecture

To motivate the proposed neural network architecture we consider a simple 1D system for which the equation is unknown. The neural network is tasked with predicting the evolution of this system. The system consists of some quantity $w(x,t)$ which evolves on a periodic domain $x \in \Omega$. We know it satisfies the following conservation laws:

$$\int_\Omega \frac{\mathrm{d}w}{\mathrm{d}t}\mathrm{d}x = 0, \tag{D.1}$$

$$\int_\Omega w\frac{\mathrm{d}w}{\mathrm{d}t}\mathrm{d}x = 0, \tag{D.2}$$

which resemble momentum and energy conservation in the Navier-Stokes equations. $w(x,t)$ is discretized on a finite volume grid, such that $w(x_i,t) \approx \mathrm{w}_i(t)$, where x_i is the center of finite volume cell Ω_i. The grid contains N grid cells such that the discrete solution is described by the state vector $\mathbf{w}(t) \in \mathbb{R}^N$. We consider the case in which the evolution equation for $w(x,t)$ is unknown; however, we do have data for $w(x,t)$ at different points in space and time. The challenge is finding the RHS for $\frac{\mathrm{d}\mathbf{w}_h}{\mathrm{d}t}$, with (D.1) and (D.2) being satisfied discretely, i.e.

$$\mathbf{1}_h^T\mathbf{\Omega}_h\frac{\mathrm{d}\mathbf{w}_h}{\mathrm{d}t} = 0, \tag{D.3}$$

$$\mathbf{w}_h^T\mathbf{\Omega}_h\frac{\mathrm{d}\mathbf{w}_h}{\mathrm{d}t} = 0, \tag{D.4}$$

are satisfied. As an ansatz for the RHS we use our proposed skew-symmetric neural network architecture

$$\mathbf{\Omega}_h\frac{\mathrm{d}\mathbf{w}_h}{\mathrm{d}t} \approx \underbrace{(\mathbf{\Delta}_c\mathrm{diag}(\mathbf{k})\mathbf{\Delta}_f - \mathbf{\Delta}_f^T\mathrm{diag}(\mathbf{k})\mathbf{\Delta}_c^T)}_{=:\mathcal{Y}}\mathbf{w}_h, \tag{D.5}$$

where $\mathbf{\Delta}_c, \mathbf{\Delta}_f \in \mathbb{R}^{N\times N}$ are central and forward difference stencils, respectively, such that $(\mathbf{\Delta}_c\mathbf{w}_h)_i = \mathrm{w}_{h,i+1} - \mathrm{w}_{h,i-1}$ and $(\mathbf{\Delta}_f\mathbf{w}_h)_i = \mathrm{w}_{h,i+1} - \mathrm{w}_{h,i}$. Note that these are specific choices for $\mathcal{B}_1$ and $\mathcal{B}_2$. During testing, we found that predefining the convolutions in $\mathcal{B}_1$ and $\mathcal{B}_2$ resulted in poor performance of the closure model. However, we choose to do so here to ease the analysis. Note that (D.3) is satisfied due to $\mathbf{\Delta}_c$ and $\mathbf{\Delta}_f$ being in the nullspace of $\mathbf{1}_h$ and (D.4) is satisfied due to the skew-symmetry of $\mathcal{Y}$. We are free to choose $\mathbf{k} \in \mathbb{R}^N$ without violating the conservation laws. In our case, it is represented by the output of a CNN, such that $\mathbf{k} = \mathbf{k}(\mathbf{w}_h, \theta) \in \mathbb{R}^N$. Writing out the matrix-vector product in

(D.5) gives us

$$\mathcal{Y}\mathbf{w}_h = \begin{bmatrix} \ddots & & \ddots & & & & \\ & -\mathrm{k}_{i-1} & \mathrm{k}_i + \mathrm{k}_{i-1} & \mathrm{k}_{i-1} & & & \\ & & \ddots & -\mathrm{k}_{i-1} - \mathrm{k}_i & \ddots & & \\ & & & 0 & -\mathrm{k}_i - \mathrm{k}_{i+1} & \mathrm{k}_{i+1} & \\ & & & \mathrm{k}_{i+1} + \mathrm{k}_i & \ddots & \ddots & \ddots \\ & & & -\mathrm{k}_{i+1} & \ddots & & \\ & & & & \ddots & & \end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ \vdots \\ \mathrm{w}_{h,i} \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}$$

$$= \begin{bmatrix} \ddots & & & & \\ & \ddots & \mathrm{w}_{h,i+1} - \mathrm{w}_{h,i} & & \\ \ddots & \mathrm{w}_{h,i-1} - \mathrm{w}_{h,i-2} & \mathrm{w}_{h,i-1} - \mathrm{w}_{h,i+1} & \mathrm{w}_{h,i+2} - \mathrm{w}_{h,i+1} & \\ & & \mathrm{w}_{h,i} - \mathrm{w}_{h,i-1} & \ddots & \ddots \\ & & & \ddots & \end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ \mathrm{k}_i \\ \vdots \\ \vdots \end{bmatrix},$$

where it is clear that for both matrices the columns lie in the nullspace of $\mathbf{1}_h$. This gives us two perspectives: the first is that of a highly parameterized skew-symmetric matrix multiplied by the solution vector $\mathbf{w}_h$. The second is that of the parameterized vector $\mathbf{k}(\mathbf{w}_h, \theta)$ being multiplied by a matrix for which each column lies in the nullspace of $\mathbf{w}_h$. Therefore, by training $\mathbf{k}(\mathbf{w}_h, \theta)$ on reference data we effectively learn a highly parameterized, and non-linear, momentum and energy conserving discretization. This formulation allows the neural network to freely explore the energy-conserving vector space. By choosing $k_i = 1/(2h)$ and $k_{i-1} = k_{i+1} = 0$, where h is the grid spacing, we simply obtain a central difference approximation of the first derivative, which corresponds to linear advection. The skew-symmetric term can therefore be viewed as highly parameterized advection.

For dissipative systems, i.e. (D.2) is negative, we require an additional term in our parameterized discretization, as introduced in Section 4.3.2. This extends (D.5) to

$$\mathbf{\Omega}_h \frac{\mathrm{d}\mathbf{w}_h}{\mathrm{d}t} \approx \mathcal{Y}\mathbf{w}_h \underbrace{-\mathbf{\Delta}_f^T \mathrm{diag}(\mathbf{q})^2 \mathbf{\Delta}_f}_{=:\mathcal{Z}} \mathbf{w}_h, \tag{D.6}$$

where $\mathbf{q} = \mathbf{q}(\mathbf{w}_h, \theta) \in \mathbb{R}^N$ is also an output of the CNN. In this case we choose $\mathcal{B}_3 = \mathbf{\Delta}_f$. This specific choice is once again made to ease the analysis, whereas in the actual neural network $\mathcal{B}_3$ is a parameterized operator. Writing out the corresponding stencil gives:

$$(\mathcal{Z}\mathbf{w}_h)_i = \mathrm{q}_i^2(\mathbf{w}_h, \theta)(w_{h,i+1} - w_{h,i}) + \mathrm{q}_{i-1}^2(\mathbf{w}_h, \theta)(w_{h,i-1} - w_{h,i}).$$

By choosing $\mathrm{q}_i = 1/h$, we obtain a second-order approximation of the second derivative. Our dissipative term can therefore be thought of as highly parameterized and non-linear diffusion.

Appendix E. Training of the neural networks

In Fig. E.14 we depict the loss of the different closure model architectures with respect to NC.

The choice of hyperparameters is discussed in Section 5.1.

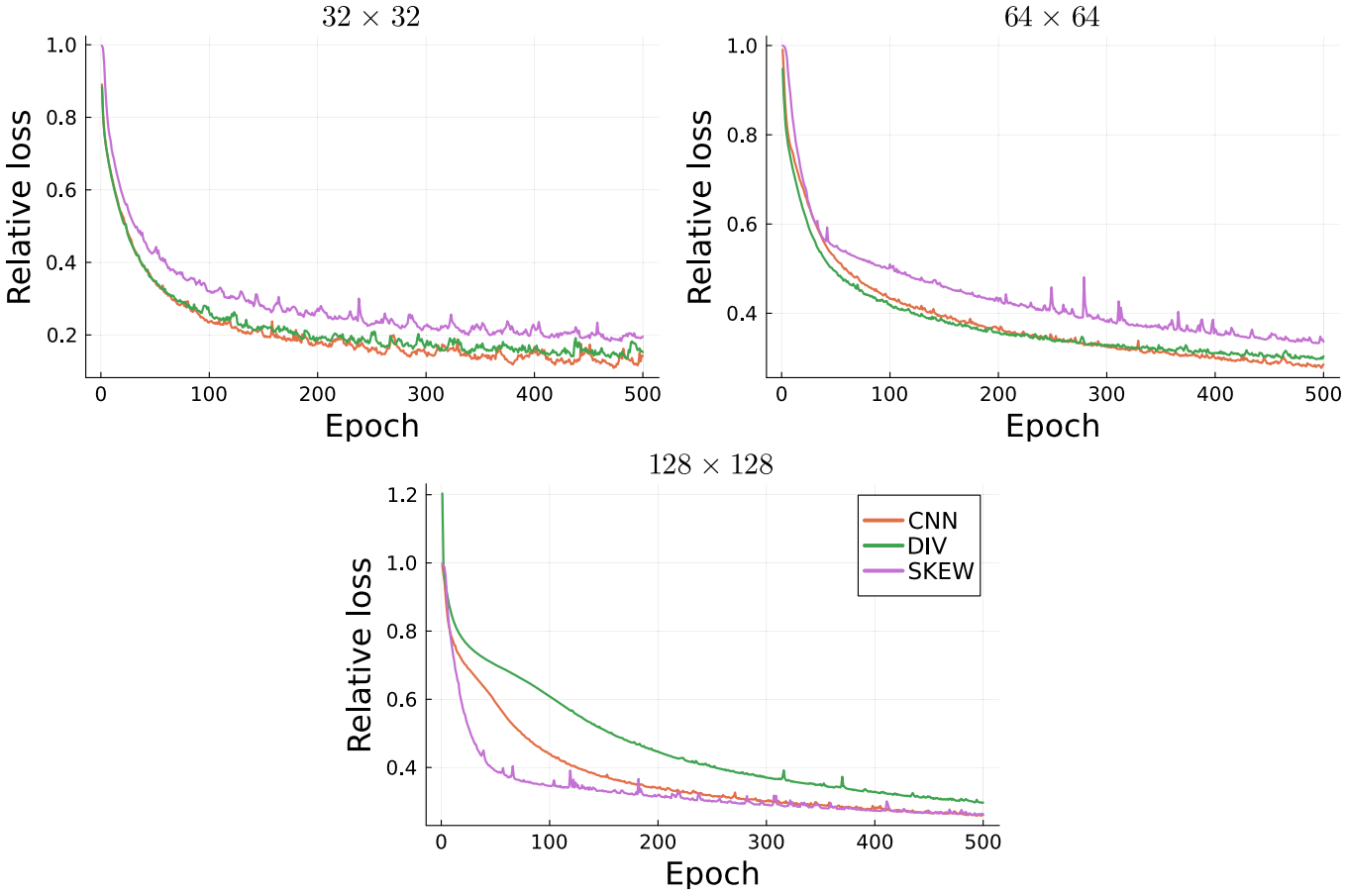

Fig. E.14. Relative loss, see (20), with respect to NC for each of the different closure models and coarse-graining factors.

Appendix F. Vorticity fields

In Figs. F.15 and F.16 we display the vorticity fields of the decaying turbulence simulation, using the trained closure models.

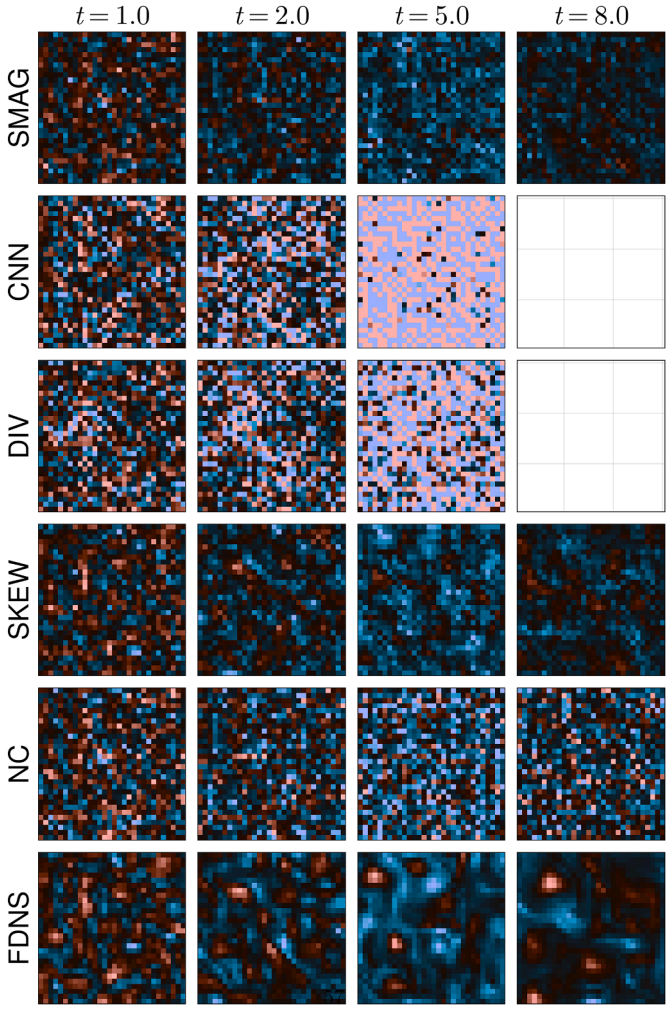

Fig. F.15. Vorticity fields at each point in time for each of the closure models on a 32×32 grid. Simulations correspond to the decaying turbulence test case. Blank boxes indicate an unstable simulation.

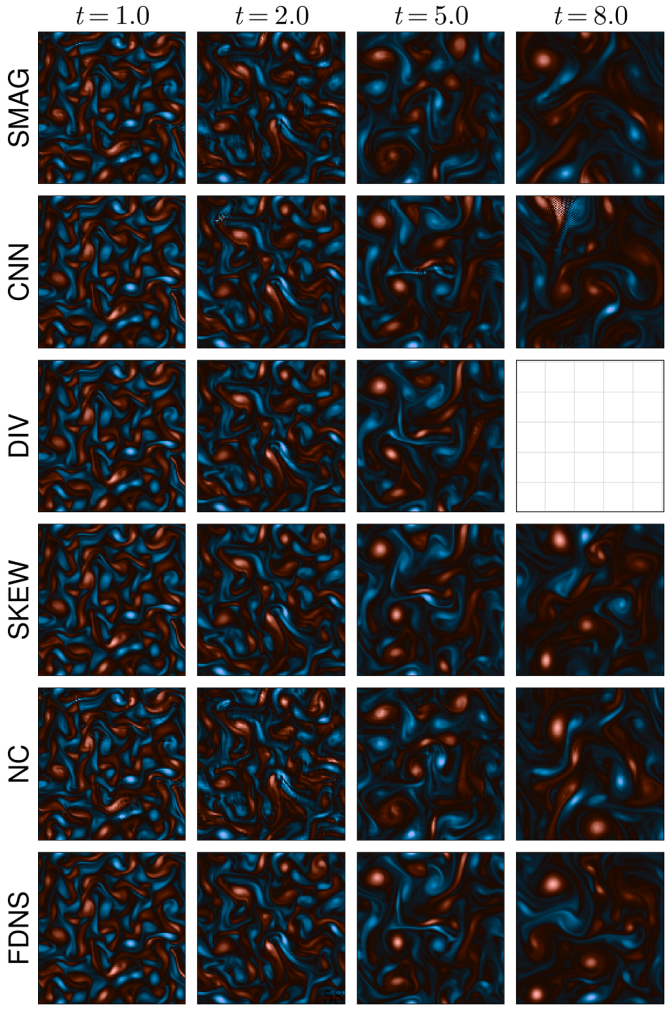

Fig. F.16. Vorticity fields at each point in time for each of the closure models on a 128×128 grid. Simulations correspond to the decaying turbulence test case. Blank boxes indicate an unstable simulation.

References

[1] Sasaki D, Obayashi S, Nakahashi K. Navier-Stokes optimization of supersonic wings with four objectives using evolutionary algorithm. J Aircr 2002;39(4):621–9.

[2] Zalaletdinov R. Averaging out inhomogeneous Newtonian cosmologies: II. Newtonian cosmology and the Navier-Stokes-Poisson equations. 2002. https://arxiv.org/abs/gr-qc/0212071.

[3] Sanderse B. Energy-conserving discretization methods for the incompressible Navier-Stokes equations: application to the simulation of wind-turbine wakes. Phd thesis 2 (research not tu/e / graduation tu/e); Centrum voor Wiskunde en Informatica; 2013. https://doi.org/10.6100/IR750543

[4] Girault V, Raviart P-A. Finite element methods for Navier-Stokes equations: theory and algorithms; vol. 5. Springer Science & Business Media; 2012.

[5] Strikwerda JC. Finite difference methods for the stokes and Navier-Stokes equations. SIAM J Sci Stat Comput 1984;5(1):56–68. https://doi.org/10.1137/0905004

[6] Harlow FH, Welch JE. Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. Phys Fluids 1965;8(12):2182.

[7] Coppola G, Veldman A EP. Global and local conservation of mass, momentum and kinetic energy in the simulation of compressible flow. J Comput Phys 2023;475:111879. https://doi.org/10.1016/j.jcp.2022.111879

[8] Verstappen R, Veldman A. Symmetry-preserving discretization of turbulent flow. J Comput Phys 2003;187:343–68. https://doi.org/10.1016/S0021-9991(03)00126-8

[9] Chu S, Kurganov A, Xin R. New Low-dissipation central-upwind schemes. Part II. 2024. https://arxiv.org/abs/2405.07620.

[10] Sagaut P, Meneveau C. Large eddy simulation for incompressible flows: an introduction. Scientific Computation; Springer; 2006. ISBN: 9783540263449. https://books.google.nl/books?id=ODYiH6RNyoQC.

[11] Agdestein SD, Sanderse B. Learning filtered discretization operators: non-intrusive versus intrusive approaches. 2022. https://doi.org/10.48550/ARXIV.2208.09363

[12] Melchers H, Crommelin D, Koren B, Menkovski V, Sanderse B. Comparison of neural closure models for discretised PDEs. arXiv:221014675 2022;.

[13] van Gastelen T, Edeling W, Sanderse B. Energy-conserving neural network for turbulence closure modeling. J Comput Phys 2024;508:113003. https://doi.org/10.1016/j.jcp.2024.113003

[14] Agdestein SD, Sanderse B. Discretize first, filter next: learning divergence-consistent closure models for large-eddy simulation. J Comput Phys 2025;522:113577. https://doi.org/10.1016/j.jcp.2024.113577

[15] Shankar V, Chakraborty D, Viswanathan V, Maulik R. Differentiable turbulence: closure as a partial differential equation constrained optimization. 2024. https://arxiv.org/abs/2307.03683.

[16] Smagorinsky J. General circulation experiments with the primitive equations: I. The basic experiment. Mon Weather Rev 1963;91(3):99–164.

[17] Pope SB. Turbulent flows. Cambridge University Press; 2000.

[18] Domaradzki JA, Metcalfe RW, Rogallo RS, Riley JJ. Analysis of subgrid-scale eddy viscosity with use of results from direct numerical simulations. Phys Rev Lett 1987;58:547–50. https://doi.org/10.1103/PhysRevLett.58.547

[19] Carati D, Ghosal S, Moin P. On the representation of backscatter in dynamic localization models. Phys Fluids 1995;7(3):606–16. https://doi.org/10.1063/1.868585

[20] Grooms I, Lee Y, Majda AJ. Numerical schemes for stochastic backscatter in the inverse cascade of quasigeostrophic turbulence. Multiscale Model Simul 2015;13(3):1001–21. https://doi.org/10.1137/140990048

[21] Meneveau C, Katz J. Scale-invariance and turbulence models for large-eddy simulation. Annu Rev Fluid Mech 2000;32:1–32. https://doi.org/10.1146/annurev.fluid.32.1.1

[22] Germano M, Piomelli U, Moin P, Cabot WH. A dynamic subgrid-scale eddy viscosity model. Phys Fluids A Fluid Dyn 1991;3(7):1760–5. https://doi.org/10.1063/1.857955

[23] Lilly DK, et al. A proposed modification of the Germano subgrid-scale closure method. Phys Fluids A Fluid Dyn 1992;4(3):633–5. https://doi.org/10.1063/1.858280

[24] Jansen M, Held I, Adcroft A, Hallberg R. Energy budget-based backscatter in an eddy permitting primitive equation model. Ocean Modell 2015;94. https://doi.org/10.1016/j.ocemod.2015.07.015

[25] Edoh A, Karagozian AR. Stabilized scale-similarity modeling for explicitly-filtered large-eddy simulations. In: 55th AIAA aerospace sciences meeting. Grapevine, Texas: American Institute of Aeronautics and Astronautics. ISBN: 978-1-62410-447-3; 2017, https://doi.org/10.2514/6.2017-1227

[26] Iyer PS, Malik MR. Efficient dynamic mixed subgrid-scale model. Phys Rev Fluids 2024;9(9):L092601. https://doi.org/10.1103/PhysRevFluids.9.L092601

[27] Shi Y, Xiao Z, Chen S. Constrained subgrid-scale stress model for large eddy simulation. Phys. Fluids 2008;20(1):011701. https://doi.org/10.1063/1.2831134

[28] Hewitt HT, Roberts M, Mathiot P, Biastoch A, Blockley E, Chassignet EP, et al. Resolving and parameterising the ocean mesoscale in earth system models. Curr Clim Change Rep 2020;6(4):137–52. https://doi.org/10.1007/s40641-020-00164-w

[29] Kurz M, Beck A. A machine learning framework for LES closure terms. ETNA - Electron Trans Numer Anal 2022;56:117–37. https://doi.org/10.1553/etna_vol56s117

[30] Kurz M, Offenhäuser P, Beck A. Deep reinforcement learning for turbulence modeling in large eddy simulations. 2022. https://doi.org/10.48550/ARXIV.2206.11038

[31] List B, Chen L-W, Thuerey N. Learned turbulence modelling with differentiable fluid solvers. 2022. https://doi.org/10.48550/ARXIV.2202.06988

[32] Maulik R, San O, Rasheed A, Vedula P. Subgrid modelling for two-dimensional turbulence using neural networks. J Fluid Mech 2018;858:122–44. https://doi.org/10.1017/jfm.2018.770

[33] Kochkov D, Smith JA, Alieva A, Wang Q, Brenner MP, Hoyer S. Machine learning – accelerated computational fluid dynamics. Proc Natl Acad Sci 2021;118(21):e2101784118. https://doi.org/10.1073/pnas.2101784118

[34] Park J, Choi H. Toward neural-network-based large eddy simulation: application to turbulent channel flow. J Fluid Mech 2021;914:A16. https://doi.org/10.1017/jfm.2020.931

[35] Boral A, Wan ZY, Zepeda-Núñez L, Lottes J, Wang Q, Chen Y-F, et al. Neural ideal large eddy simulation: modeling turbulence with neural stochastic differential equations. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S, editors. Advances in neural information processing systems; vol. 36. Curran Associates, Inc.; 2023, p. 69270–83. https://proceedings.neurips.cc/paper_files/paper/2023/file/dabaded617b3be96c3ed161498a7d71c-Paper-Conference.pdf.

[36] Guan Y, Chattopadhyay A, Subel A, Hassanzadeh P. Stable a posteriori LES of 2D turbulence using convolutional neural networks: backscattering analysis and generalization to higher Re via transfer learning. J Comput Phys 2022;458:111090. https://doi.org/10.1016/j.jcp.2022.111090

[37] Beck A, Flad D, Munz C-D. Deep neural networks for data-driven LES closure models. J Comput Phys 2019;398:108910. https://doi.org/10.1016/j.jcp.2019.108910

[38] Kurz M, Beck A. Investigating model-data inconsistency in data-informed turbulence closure terms. In: 14th WCCM-ECCOMAS congress 2020. 2021, https://doi.org/10.23967/wccm-eccomas.2020.115

[39] Frezat H, Sommer JL, Fablet R, Balarac G, Lguensat R. A posteriori learning of quasi-geostrophic turbulence parametrization: an experiment on integration steps. 2021. https://doi.org/10.48550/ARXIV.2111.06841

[40] MacArt JF, Sirignano J, Freund JB. Embedded training of neural-network subgrid-scale turbulence models. Phys Rev Fluids 2021;6:050502. https://doi.org/10.1103/PhysRevFluids.6.050502

[41] Bae HJ, Koumoutsakos P. Scientific multi-agent reinforcement learning for wall-models of turbulent flows. Nat Commun 2022;13(1):1443. https://doi.org/10.1038/s41467-022-28957-7

[42] Sanderse B, Stinis P, Maulik R, Ahmed SE. Scientific machine learning for closure models in multiscale problems: a review. 2024. 2403.02913; https://arxiv.org/abs/2403.02913.

[43] Prakash A, Jansen KE, Evans JA. Invariant data-driven subgrid stress modeling in the strain-rate eigenframe for large eddy simulation. Comput Methods Appl Mech Eng 2022;399:115457. https://doi.org/10.1016/j.cma.2022.115457
[44] Hernandez Q, Badias A, Chinesta F, Cueto E. Thermodynamics-informed graph neural networks. IEEE Trans Artif Intell 2022;:1–1. https://doi.org/10.1109/tai.2022.3179681
[45] Sanderse B. Non-linearly stable reduced-order models for incompressible flow with energy-conserving finite volume methods. J Comput Phys 2020;421:109736. https://doi.org/10.1016/j.jcp.2020.109736
[46] de Moura CA, Kubrusly CS. The Courant–Friedrichs–Lewy (CFL) Condition: 80 Years After Its Discovery. Birkhäuser, 2013.
[47] Kingma DP, Ba J. Adam: a method for stochastic optimization. 2014. https://doi.org/10.48550/ARXIV.1412.6980
[48] Glorot X, Bordes A, Bengio Y. Deep sparse rectifier neural networks. vol. 15. 2010.
[49] Butcher J. Numerical methods for ordinary differential equations. John Wiley & Sons, Ltd; 2016. ISBN: 9781119121503. https://doi.org/10.1002/9781119121534
[50] Horn RA, Johnson CR. Matrix analysis. Cambridge University Press; 2nd ed.; 2012.
[51] Bauschke HH, Wang X, Yao L. On Borwein-Wiersma decompositions of monotone linear relations. 2009. https://arxiv.org/abs/0912.2772.
[52] Gallavotti G. Foundations of fluid dynamics. Berlin, Heidelberg: Springer; 2002. ISBN: 9783540426345. https://doi.org/10.1007/978-3-662-04968-2
[53] Bank D, Koenigstein N, Giryes R. Autoencoders. Machine learning for data science handbook: data mining and knowledge discovery handbook 2023;:353–74.
[54] Butcher J. Runge-Kutta methods. Scholarpedia 2007;2(9):3147. Revision # 91735. https://doi.org/10.4249/scholarpedia.3147
[55] Sanderse B, Koren B. Accuracy analysis of explicit Runge-Kutta methods applied to the incompressible Navier-Stokes equations. J Comput Phys 2012;231(8):3041–63. https://doi.org/10.1016/j.jcp.2011.11.028
[56] Poinsot T, Candel SM. The influence of differencing and CFL number on implicit time-dependent non-linear calculations. J Comput Phys 1986;62(2):282–96. https://doi.org/10.1016/0021-9991(86)90128-2
[57] Bezanson J, Edelman A, Karpinski S, Shah VB. Julia: a fresh approach to numerical computing. SIAM Rev 2017;59(1):65–98. 10.1137/141000671.
[58] Innes M, Saba E, Fischer K, Gandhi D, Rudilosso MC, Joy NM, et al. Fashionable modelling with flux. CoRR 2018; https://arxiv.org/abs/1811.01457.
[59] Innes M. Flux: elegant machine learning with Julia. J Open Source Softw 2018; https://doi.org/10.21105/joss.00602
[60] SURF. Snellius: the national supercomputer. Accessed: 2025-03-21; https://www.surf.nl/en/services/snellius-the-national-supercomputer.
[61] Gatski TB, Hussaini MY, Lumley JL. Simulation and modeling of turbulent flows. Oxford University Press; 1996. ISBN: 9780195106435. https://doi.org/10.1093/oso/9780195106435.001.0001
[62] Canuto VM, Cheng Y, et al. Determination of the Smagorinsky-Lilly constant CS. Phys Fluids 1997;9(5):1368–78. https://doi.org/10.1063/1.869251
[63] Bartosiewicz Y, Duponcheel M. 6.1.2 - Large-eddy simulation: application to liquid metal fluid flow and heat transfer. In: Roelofs F, editor. Thermal hydraulics aspects of liquid metal cooled nuclear reactors. Woodhead Publishing. ISBN: 978-0-08-101980-1; 2019, p. 245–71. https://doi.org/10.1016/B978-0-08-101980-1.00017-X
[64] Kraichnan RH. Inertial ranges in two-dimensional turbulence. Phys Fluids 1967;10(7):1417–23. https://doi.org/10.1063/1.1762301
[65] Feng ZC, Li YC. Short-term unpredictability of high Reynolds number turbulence – rough dependence on initial data. 2017;Preprint available at https://arxiv.org/abs/1702.02993
[66] Coveney PV, Dougherty ER, Highfield RR. Big data need big theory too. Philos Trans R Soc A Math Phys Eng Sci 2016;374(2080):20160153.
[67] De Avila B-PF, Economon T, Kolter Z. Combining differentiable PDE solvers and graph neural networks for fluid flow prediction. In: International conference on machine learning. PMLR; 2020, p. 2402–11.