Discretize first, filter next: Learning divergence-consistent closure models for large-eddy simulation

Syver Døving Agdestein [a,b,*], Benjamin Sanderse [a,b]

[a] *Scientific Computing Group, Centrum Wiskunde & Informatica, Science Park 123, Amsterdam, 1098 XG, the Netherlands*
[b] *Centre for Analysis, Scientific Computing and Applications, Eindhoven University of Technology, PO Box 513, Eindhoven, 5600 MB, the Netherlands*

A R T I C L E   I N F O

A B S T R A C T

We propose a new neural network based large eddy simulation framework for the incompressible Navier-Stokes equations based on the paradigm "discretize first, filter and close next". This leads to full model-data consistency and allows for employing neural closure models in the same environment as where they have been trained. Since the LES discretization error is included in the learning process, the closure models can learn to account for the discretization.

Furthermore, we employ a divergence-consistent discrete filter defined through face-averaging and provide novel theoretical and numerical filter analysis. This filter preserves the discrete divergence-free constraint by construction, unlike general discrete filters such as volume-averaging filters. We show that using a divergence-consistent LES formulation coupled with a convolutional neural closure model produces stable and accurate results for both a-priori and a-posteriori training, while a general (divergence-inconsistent) LES model requires a-posteriori training or other stability-enforcing measures.

1. Introduction

The incompressible Navier-Stokes equations form a model for the movement of fluids. They can be solved numerically on a grid using discretization techniques such as finite differences [29], finite volumes, or pseudo-spectral methods [53,58]. The important dimensionless parameter in the incompressible Navier-Stokes equations is the Reynolds number $\mathrm{Re} = \frac{UL}{\nu}$, where L is a characteristic length scale, U is a characteristic velocity, and ν the kinematic viscosity. For high Reynolds numbers, the flow becomes turbulent. Resolving all the scales of motion of turbulent flows requires highly refined computational grids. This is computationally expensive [57,60,59,76].

Large eddy simulation (LES) aims to resolve only the large scale features of the flow, as opposed to direct numerical simulation (DNS), where all the scales are resolved [57,8]. The large scales of the flow, here denoted $\bar{u}$, are extracted from the full solution u using a spatial filter. The equations for the large scales are then obtained by filtering the Navier-Stokes equations. The large scale equations (filtered DNS equations) are not closed, as they still contain terms depending on the small scales. It is common to group the contributions of the unresolved scales into a single commutator error term, that we denote $c(u)$. Large eddy simulation requires modeling this term as a function of the large scales only. The common approach is to introduce a closure model $m(\bar{u}, \theta) \approx c(u)$ to remove the small scale dependency [57,60,8], where θ are problem-specific model parameters. The closure model accounts for the

---

* Corresponding author at: Scientific Computing Group, Centrum Wiskunde & Informatica, Science Park 123, Amsterdam, 1098 XG, the Netherlands.
E-mail addresses: sda@cwi.nl (S.D. Agdestein), b.sanderse@cwi.nl (B. Sanderse).
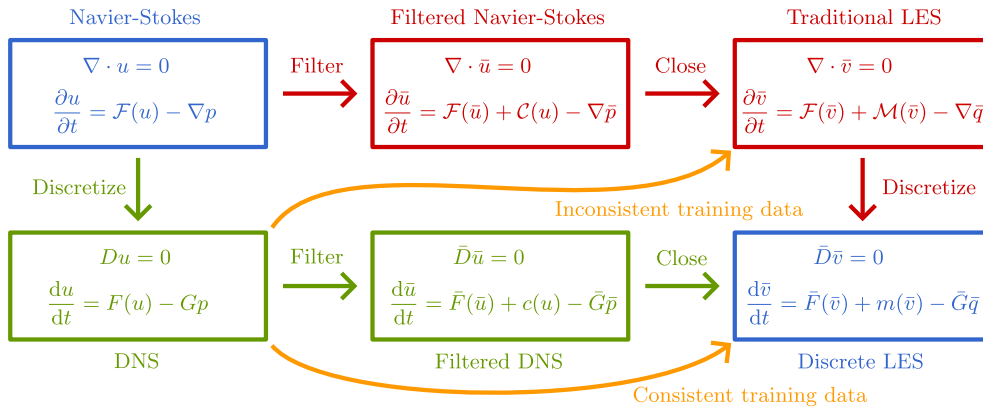
Fig. 1. Proposed route (in green) to a discrete LES model, based on "discretize first, then filter" instead of "filter first, then discretize" (in red). The term $\mathcal{F}(u)$ contains the convective and diffusive terms. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

effect of the sub-filter scales on the resolved scales. Traditional closure models account for the energy lost to the sub-filter eddies. A simple approach to account for this energy transfer is to add an additional diffusive term to the LES equations. These closure models are known as *eddy viscosity* models. This includes the well known standard Smagorinsky model [69,39,40].

The filter can be either be known explicitly or implicitly. In the latter case, the (coarse) discretization itself is acting as a filter. Note that the distinction explicit versus implicit is not always clear, as an explicitly defined filter can still be linked to the discretization. Sometimes the filter is only considered to be explicit when the filter width is fully independent from the LES discretization size [6]. The type of filtering employed results in a distinction between explicit and implicit LES. *Explicit LES* [46,21,22] refers to a technique where the filter is explicitly used inside the convection term of the LES equations (as opposed to applying the filter in the exact filtered DNS evolution equations, where the filter is applied by definition). The LES solution (that we denote by $\bar{v}$) already represents a filtered quantity, namely the filtered DNS solution $\bar{u}$. Applying the filter *again* inside the LES equations can thus be seen as computing a quantity that is filtered twice. The main advantage is stability [7] and to prevent the growth of high-wavenumber components [46,21]. Note that this technique requires explicit access to the underlying filter. *Implicit LES* is used to describe the procedure where the DNS equations are used in their original form, and the closure model is added as a correction (regardless of whether the filter is known explicitly or not). Here, the filter itself is not required to evaluate the LES equations, but knowledge of certain filter properties such as the filter width is used to choose the closure model. In the standard Smagorinsky model for example, the filter width is used as a model parameter. If the exact filter is not known, this parameter is typically chosen to be proportional to the grid spacing.

Recently, machine learning has been used to learn closure models, focusing mostly on implicit LES [4–6,20,30,38,37,43,50,68,66, 67]. The idea is to represent the closure model $m(\bar{u}, \theta)$ by an artificial neural network (ANN). ANNs are in principle a good candidate as they are universal function approximators [3,14]. However, using ANNs as closure models typically suffers from stability issues, which have been attributed to a so-called model-data inconsistency: the environment in which the neural network is trained is not the same in which it is being used [6]. Several approaches like backscatter clipping, a-posteriori training and projection onto an eddy-viscosity basis have been used to enforce stability [55,35,43,4] – for an overview, see [63]. Our view is that *one of the problems that lies at the root of the model-data inconsistency is a discrepancy between the LES equations (obtained by first filtering the continuous Navier-Stokes equations, then discretizing) and the training data (obtained by discretizing the Navier-Stokes equations, and then applying a discrete filter).*

Our key insight is that the LES equations can also be obtained by "discretizing first" instead of "filtering first": following the green instead of the red route in Fig. 1. In other words, by discretizing the PDE first, and then applying a *discrete* filter, the model-data inconsistency issue can be avoided and one can generate exact training data for the discrete LES equations. Training data obtained by filtering discrete DNS solutions is fully consistent with the environment where the discrete closure model is used. The resulting LES equations do not have a coarse grid discretization error, but an underlying fine grid discretization error and a commutator error from the discrete filter, which can be learned using a neural network. In our recent work [1], we showed the benefits of the "discretize first" approach on a 1D convection equation. With the discretize-first approach we obtain stable results without the need for the stabilizing techniques mentioned above (backscatter clipping, a-posteriori training, projection onto an eddy-viscosity basis).

In the current paper, the goal is to extend the "discretize first" approach to the full 3D incompressible Navier-Stokes equations. One major challenge that appears in incompressible Navier-Stokes is the presence of the divergence-free constraint. We show that discrete filters are in general not divergence-consistent (meaning that divergence-free DNS solutions do not stay divergence-free upon filtering). Kochkov et al. used a face-averaging filter to achieve divergence-consistency [35]. We employ this filter and show how it leads to a different set of equations than for non-divergence-consistent filters. Overall, the main result is that our divergence-consistent neural closure models lead to stable simulations.

In addition, we remark that divergence-consistent filtering is an important step towards LES closure models that satisfy an energy inequality. Such models were developed by us in [74] for one-dimensional equations with quadratic nonlinearity (Burgers, Korteweg - de Vries). When extending this approach to 3D LES, the derivation of the energy inequality hinges on having a divergence-free constraint on the filtered solution field.
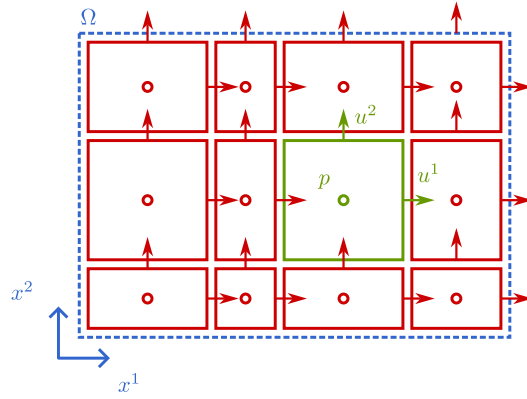
Fig. 2. Finite volume discretization on a staggered grid. The pressure is defined in the volume center, and the velocity components on the volume faces.

Our article is structured as follows. In section 2, we present the discrete DNS equations that serve as the ground truth in our problem, based on a second order accurate finite volume discretization on a staggered grid. In section 3, we introduce discrete filtering. Unclosed equations for the large scales are obtained. We show that the filtered velocity is not automatically divergence-free. We then present two discrete filters on a staggered grid, of which one is divergence-consistent. In section 4, we present our discrete closure modeling framework, resulting in two discrete LES formulations. We discuss the validity of the LES models in terms of divergence-consistency. A discussion follows on the choice of closure model and how we can learn the closure model parameters. In section 5, we present the results of numerical experiments on a decaying turbulence test case. Section 6 ends with concluding remarks.

Additional details are included in the appendices. In Appendix A, we include details about the discretization procedure. In Appendix B, we explain the numerical experiments. In Appendix C, we show that the face-averaging filter is divergence-consistent on both uniform and non-uniform grids. In Appendix D, we show the problems that can occur when filtering before differentiating the divergence-free constraint. In Appendix E, we analyze the transfer functions of a continuous face-averaging filter and a continuous volume-averaging filter. In Appendix F, we analyze the difference between "discretize first" and "filter first" for an analytical solution to the incompressible Navier-Stokes equations. In Appendix G, we provide results for two more test cases: a decaying turbulence test case in 2D, and a forced turbulence test case in 3D.

2. Direct numerical simulation of all scales

In this section, we present the continuous Navier-Stokes equations and define a discretization aimed at resolving all the scales of motion. The resulting discrete equations will serve as the ground truth for learning an equation for the large scales.

2.1. The Navier-Stokes equations

The incompressible Navier-Stokes equations describe conservation of mass and conservation of momentum, which can be written as a divergence-free constraint and an evolution equation:

$$\nabla \cdot u = 0, \tag{1}$$

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

where $\Omega \subset \mathbb{R}^d$ is the domain, $d \in \{2, 3\}$ is the spatial dimension, $u = (u^1, \ldots, u^d)$ is the velocity field, p is the pressure, ν is the kinematic viscosity, and $f = (f^1, \ldots, f^d)$ is the body force per unit of volume. The velocity, pressure, and body force are functions of the spatial coordinate $x = (x^1, \ldots, x^d)$ and time t. For the remainder of this work, we assume that Ω is a rectangular domain with periodic boundaries, and that f is constant in time.

2.2. Spatial discretization

For the discretization scheme, we use a staggered Cartesian grid as proposed by Harlow and Welch [29]. Staggered grids have excellent conservation properties [41,56], and in particular their exact discrete divergence-free constraint is important for this work. Details about the discretization can be found in Appendix A.

We partition the domain Ω into N finite volumes. Let $u(t) \in \mathbb{R}^{dN}$ and $p(t) \in \mathbb{R}^N$ be vectors containing the unknown velocity and pressure components in their canonical positions as shown in Fig. 2. They are not to be confused with their space-continuous counterparts $u(x, t)$ and $p(x, t)$, which will no longer be referred to in what follows. The discrete and continuous versions of u and p have the same physical dimensions.

Equations (1) and (2) are discretized as

$$Du = 0, \tag{3}$$

$$\frac{du}{dt} = F(u) - Gp, \tag{4}$$

where $D \in \mathbb{R}^{N \times dN}$ is the divergence operator, $G = -\Omega_u^{-1} D^{\mathsf{T}} \Omega_p \in \mathbb{R}^{dN \times N}$ is the gradient operator, $\Omega_u \in \mathbb{R}^{dN \times dN}$ and $\Omega_p \in \mathbb{R}^{N \times N}$ are element-wise scaling operators containing the velocity and pressure volume sizes, and $F(u) \in \mathbb{R}^{dN}$ contains the convective, diffusive, and body force terms.

2.3. Pressure projection

The two vector equations (3) and (4) form an index-2 differential-algebraic equation system [27,28], consisting of a divergence-free constraint and an evolution equation. Given u, the pressure can be obtained by solving the discrete Poisson equation $0 = DF(u) - DGp$, which is obtained by differentiating the divergence-free constraint in time. This can also be written

$$Lp = \Omega_p DF(u), \tag{5}$$

where the Laplace matrix $L = \Omega_p DG = -\Omega_p D\Omega_u^{-1} D^{\mathsf{T}} \Omega_p$ is symmetric and negative semi-definite. We denote by $L^\dagger$ the solver to the scaled pressure Poisson equation (5). Since no boundary value for the pressure is prescribed, L is rank-1 deficient and the pressure is only determined up to a constant. We set this constant to zero by choosing

$$L^\dagger = \begin{pmatrix} I & 0 \end{pmatrix} \begin{pmatrix} L & e \\ e^{\mathsf{T}} & 0 \end{pmatrix}^{-1} \begin{pmatrix} I \\ 0 \end{pmatrix}, \tag{6}$$

where $e = (1, \ldots, 1) \in \mathbb{R}^N$ is a vector of ones and the additional degree of freedom enforces the constraint of an average pressure of zero (i.e. $e^{\mathsf{T}} L^\dagger = 0$) [45]. In this case we still have $LL^\dagger = I$, even though $L^\dagger L \neq I$.

We now introduce the projection operator P, which plays an important role in the development of our new closure model strategy in section 3: $P = I - GL^\dagger \Omega_p D$. It is used to make velocity fields divergence-free [16], since $DP = D - \Omega_p^{-1} LL^\dagger \Omega_p D = 0$. It naturally follows that P is a projector, since $P^2 = P - GL^\dagger \Omega_p DP = P$.

Having defined P, it is (at least formally) possible to eliminate the pressure from equations (3)-(4) into a single "pressure-free" evolution equation for the velocity [62], given by

$$\frac{du}{dt} = PF(u). \tag{7}$$

This way, an initially divergence-free velocity field u stays divergence-free regardless of what forcing term F is applied. We note that equation (7) alone does not enforce the divergence-free constraint, as it also requires the initial conditions to be divergence-free.

$L^\dagger$ and P are non-local operators that are not explicitly assembled, but their action on vector fields is computed on demand using an appropriate linear solver. Formulation (7) is used as a starting point for developing a new filtering technique in section 3.

2.4. Time discretization

We use Wray's low storage third order Runge-Kutta method (Wray3) for the incompressible Navier-Stokes equations [77,62]. While explicit methods may require smaller time steps (depending on the problem-specific trade-off of stability versus accuracy), they are easier to differentiate with automatic differentiation tools.

Given the solution u_n at a time t_n, the next solution at a time t_{n+1} is given by

$$u_{n+1} = u_n + \Delta t_n \sum_{i=1}^{s} b_i k_i, \tag{8}$$

where

$$k_i = PF\left(u_n + \Delta t_n \sum_{j=1}^{i-1} a_{ij} k_j\right), \tag{9}$$

$\Delta t_n = t_{n+1} - t_n$, s is the number of stages, $a \in \mathbb{R}^{s \times s}$, and $b \in \mathbb{R}^s$. In practice, each of the RK steps (9) are performed by first computing a tentative (non-divergence-free) velocity field, subsequently solving a pressure Poisson equation, and then correcting the velocity field to be divergence-free. As the method is explicit, this is equivalent to (9) and does not introduce a splitting error [62]. For Wray3, we set $s = 3$, $b_1 = 1/4$, $b_2 = 0$, $b_3 = 3/4$, $a_{21} = 8/15$, $a_{31} = 1/4$, $a_{32} = 5/12$, and the other coefficients $a_{ij} = 0$.

3. Discrete filtering and divergence-consistency

The DNS discretization presented in the previous section is in general too expensive to simulate for problems of practical interest, and it is only used to generate reference data for a limited number of test cases. In this section, we present discrete filtering from
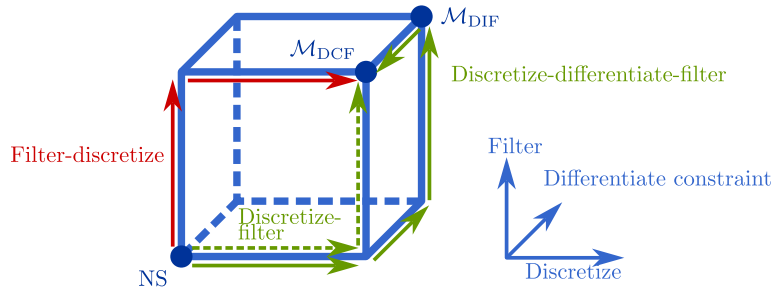
Fig. 3. Alternative view of Fig. 1 to highlight the effect of differentiating the constraint. The red arrows show the traditional route of filtering first and then discretizing. The solid green arrows show our proposed route of discretizing first, then differentiating the constraint, then filtering, and finally reintroducing a pressure term (if the filter is divergence-consistent). This is done to circumvent the pressure problems of the dashed green route (discretizing first, then filtering).

the fine DNS grid to a coarse grid in order to alleviate the computational burden. This means we filter the discretized equations, in contrast to many existing approaches, which filter the *continuous* Navier-Stokes equations, and then apply a discretization. The advantages of "discretizing first" were already mentioned in section 1. However, one disadvantage of "discretizing first" is that the filtered velocity field is in general not divergence free. Kochkov et al. pointed out that using a face-averaging filter preserves the divergence-free constraint for the filtered velocity [35]. We will employ this filter and compare it to a non-divergence-preserving volume-averaging filter.

3.1. Filtering from fine to coarse grids

We consider two computational grids: a fine grid of size $N \in \mathbb{N}^d$ and a coarse grid of size $\bar{N} \in \mathbb{N}^d$, with $\bar{N}_\alpha \leq N_\alpha$ for all $\alpha \in \{1, \ldots, d\}$. The operators D, F, G, P, etc., are defined on the fine grid. On the coarse grid, similar operators are denoted $\bar{D}$, $\bar{F}$, $\bar{G}$, $\bar{P}$, etc.

Consider a flow problem. We assume that the flow is fully resolved on the fine grid, meaning that the grid spacing is at least twice as small as the smallest significant spatial structure of the flow. The resulting fully resolved solution $u \in \mathbb{R}^{dN}$ is referred to as the *DNS solution*. In addition, we assume that the flow is not fully resolved on the coarse grid, meaning that the coarse grid spacing is larger than the smallest significant spatial structure of the flow. The aim is to solve for the large scale features of u on the coarse grid. For this purpose, we construct a discrete spatial filter $\Phi \in \mathbb{R}^{d\bar{N} \times dN}$. The resulting filtered DNS velocity field is given by

$$\bar{u} = \Phi u \in \mathbb{R}^{d\bar{N}}, \tag{10}$$

and is a coarse-grid quantity. We stress that $\bar{u}$ is by definition a consequence of the DNS. It is *not* obtained by solving the Navier-Stokes equations on the coarse grid. That is instead the goal in the next sections.

Since Φ is a coarse-graining filter, it does not generally commute with discrete differential operators. In particular, the divergence-free constraint is preserved for continuous convolutional filters, but this is not automatically the case for discrete filters. We consider this property in detail, and investigate its impact on the resulting large scale equations.

3.2. Equation for large scales

When directly filtering the differential-algebraic system (3)-(4), multiple challenges arise. These are detailed in Appendix D and summarized here:

- The filter Φ works on inputs defined in the velocity points, and is targeted at filtering the momentum equation. It is not directly clear how to filter the divergence-free constraint (which is defined in the pressure points), and whether a second filter needs to be defined for the pressure points.
- The momentum equation includes a pressure term. While its gradient can be filtered with Φ, it is not clear what a filtered pressure $\bar{p}$ should be, or how it should appear in the filtered momentum equation.

To circumvent these issues, we propose to differentiate the discrete divergence constraint first (to remove the pressure), and then apply the filter to the pressure-free DNS equation (7). This results in the sequence "discretize – differentiate constraint – filter", as shown by solid green arrows in Fig. 3. The advantage over the route defined by dashed green arrows, "discretize – filter", is that we do not need to consider the pressure or the divergence-free constraint, and a single filter for the velocity field is sufficient. The "implied" filtered pressure will be discussed in section 4.1. The resulting equation for the filtered DNS-velocity $\bar{u}$ is $\frac{\mathrm{d}\bar{u}}{\mathrm{d}t} = \Phi P F(u)$, which is rewritten as

$$\frac{\mathrm{d}\bar{u}}{\mathrm{d}t} = \bar{P}\bar{F}(\bar{u}) + c(u), \tag{11}$$

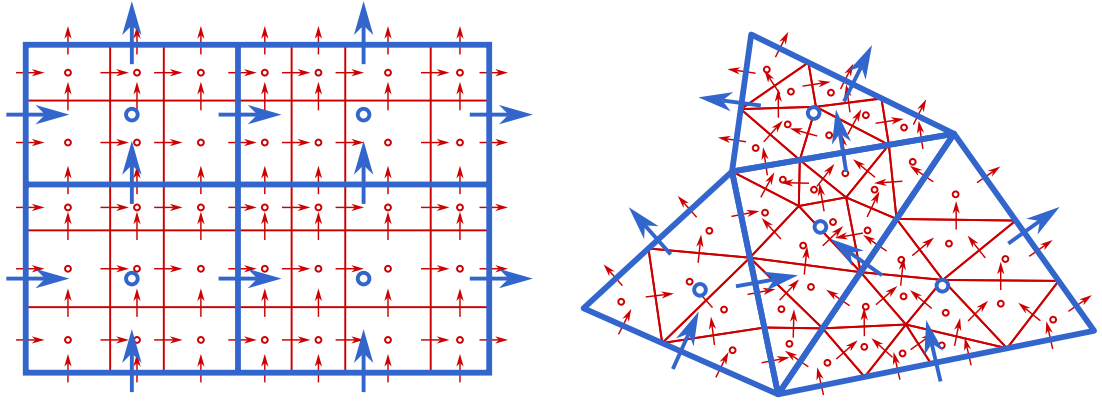with the unclosed commutator error defined by

Fig. 4. Four coarse volumes (blue) and their fine grid sub-grid volumes (red) in 2D. For each of the coarse volume faces, the discrete filter Φ^{FA} combines the DNS velocities u into one LES velocity $\bar{u}$ using averaging. The interior sub-grid velocities are not present in $\bar{u}$. The coarse grid pressure $\bar{p}$ is defined in the coarse volume centers, but is not obtained by filtering p. Instead, it is computed from $\bar{u}$. **Left:** Structured grid, used in this work. **Right:** Unstructured grid.
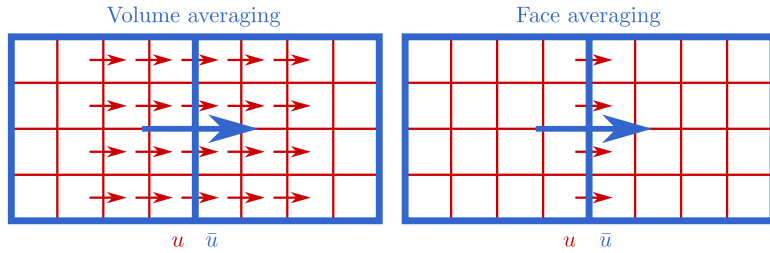

Fig. 5. DNS velocity components u contributing to a single filtered DNS component $\bar{u}$ for two filters. Both filters have a filter width equal to the grid size. Both $u(t)$ and $\bar{u}(t)$ share the same dimension as the continuous velocity $u(x, t)$. **Left:** Volume-averaging filter Φ^{VA}. **Right:** Face-averaging filter Φ^{FA}.

$$c(u) = \Phi P F(u) - \bar{P}\bar{F}(\Phi u). \tag{12}$$

A crucial point is that when filtering from a fine grid to a coarse grid, one generally does *not* get a divergence-free filtered velocity field ($\bar{D}\bar{u} \neq 0$), as discrete filtering and discrete differentiation do not generally commute.

3.3. Divergence-consistent discrete filter

Our approach is to design the filter and coarse grid such that the divergence-free constraint is preserved. This is achieved by merging fine DNS volumes to form coarse LES volumes, such that the faces of DNS and LES volumes overlap (as shown in Fig. 4). The extracted large scale velocities $\bar{u} = \Phi u$ are then obtained by averaging the DNS velocities that are found on the LES volume *faces* [35]. We denote this face-averaging discrete filter by Φ^{FA}. This approach to filtering also naturally generalizes to unstructured grids, as long as the coarse volume faces overlap with the fine ones. An example is shown to the right in Fig. 4 for triangular volumes.

The face-averaging filter Φ^{FA} is different to more traditional volume-averaging filters such as the volume-averaging top-hat filter [60] (that we denote Φ^{VA}). The face-averaging filter can be thought of as a top-hat filter acting on the dimensions orthogonal to the velocity components only, while the volume-averaging filter is averaging over all dimensions. The associated transfer function of these two types of filters are further analyzed in Appendix E. In this work, we define the two discrete filters Φ^{FA} and Φ^{VA} with uniform weights and with filter width equal to the coarse grid spacing $\bar{\Delta}$. Since both filters are top-hat like, the filter width is defined as the diameter of the averaging domain in the infinity norm. Note that due to the normalization, all discrete velocity components $\bar{u}_J^\alpha$ and u_I^α share the same dimension as the continuous velocity $u^\alpha(x, t)$ (not velocity times area or velocity times volume).

The two discrete filter supports are compared in Fig. 5. One advantage of the face-averaging filter is that it does not require modifications at non-periodic boundaries, while the volume-averaging filter does (for example by using a volume of half the size and twice as large weights to avoid averaging outside solid walls). But the main advantage lies in the preservation of the divergence-free constraint, as we now show.

When $\bar{u}$ is obtained through face-averaging, the difference of fluxes entering and leaving an LES volume is equal to a telescoping sum of all sub-grid flux differences, which in turn is zero due to the fine grid divergence-free constraint. The proof is shown in Appendix C. Note that this property no longer holds if the filter weights are non-uniform or if the filter width is different from the coarse grid spacing. The face-averaging filter thus preserves *by construction* the divergence-free constraint for the filtered DNS velocity:

$$\text{For } \Phi^{\mathrm{FA}}: \quad \boxed{\forall u, \; Du = 0 \implies \bar{D}\bar{u} = 0.} \tag{13}$$

This property is the primary motivation for our filter choice (it can also be thought of as a discrete equivalent of the divergence theorem $\nabla \cdot u = 0 \implies \int_{\partial \mathcal{O}} u \cdot n \, d\Gamma = 0$ for all $\mathcal{O}$). It can be used to show that the face-averaging filter has the property

$$\bar{D}\Phi P = 0 \tag{14}$$

since, for all u, if $w = Pu$, then $Dw = 0$ by definition of P, and thus $\bar{D}\Phi Pu = \bar{D}\Phi w = \bar{D}\bar{w} = 0$ from (13). In other words, divergence-free fine grid velocity fields stay divergence-free upon filtering. For a general volume-averaging filter, we would not be able to guarantee that all the sub-grid fluxes cancel out, and we would not get a divergence-free constraint for $\bar{u}$. This constraint is often enforced anyways, possibly introducing unforeseen errors, as pointed out by Sirignano et al. [68]. With the face-averaging filter, we do not need to worry about such errors.

With the face-averaging filter choice, the momentum commutator error is divergence-free, since

$$\bar{D}c = \bar{D}\left(\frac{d\bar{u}}{dt} - \bar{P}\bar{F}(\bar{u})\right) = \frac{d(\bar{D}\bar{u})}{dt} - (\bar{D}\bar{P})\bar{F}(\bar{u}) = 0 \tag{15}$$

since $\bar{D}\bar{u} = 0$ and $\bar{D}\bar{P} = 0$ on the coarse grid just like $DP = 0$ on the fine grid. As a result, $c = \bar{P}c$, and the right hand side of the large scale equation is divergence-free. This allows us to rewrite equation (11) as

$$\boxed{\frac{d\bar{u}}{dt} = \bar{P}\left(\bar{F}(\bar{u}) + c(u)\right).} \tag{16}$$

The fact that the projection operator $\bar{P}$ now also acts on the commutator error will play an important role in learning a new LES closure model in section 4.

Since $Du(0) = 0$ and thus $\bar{D}\bar{u}(0) = 0$ (for the face-averaging filter), we can rewrite the filtered equation into an equivalent constrained form, similar to the unfiltered equations (3)-(4):

$$\bar{D}\bar{u} = 0, \tag{17}$$

$$\frac{d\bar{u}}{dt} = \bar{F}(\bar{u}) + c(u) - \bar{G}\bar{p}, \tag{18}$$

where $\bar{p}$ is the "implied" pressure defined in the coarse volume centers. It is obtained by solving the pressure Poisson equation with the additional sub-grid forcing term c in the right hand side:

$$\bar{L}\bar{p} = \bar{\Omega}_p \bar{D}\left(\bar{F}(\bar{u}) + c(u)\right). \tag{19}$$

Note that the coarse pressure $\bar{p}$ is not obtained defining a pressure filter, but arises from enforcing the coarse grid divergence-free constraint. In other words, by filtering the pressure-free momentum equation (7) with a divergence-consistent filter, we discover what the (implicitly defined) filtered pressure is. An implicit volume-averaging pressure filter can still be defined, see Appendix C for further details.

3.4. Other divergence-consistent filters

We comment here on other approaches for constructing divergence-consistent discrete filters.

3.4.1. Discrete differential filters

Continuous filters can be built using differential operators, for example Germano's filter $\bar{u}(x,t) = (1 - \bar{\Delta}^2/24\nabla^2)^{-1}u$ [24,12]. Differential filters can also be extended to the discrete case. Trias and Verstappen propose using polynomials of the discrete diffusion operator (which we will denote D_2) as a filter [72]:

$$\Phi = I + \sum_{i=1}^{m} \gamma_i D_2^i, \tag{20}$$

where m is the polynomial degree and γ_i are filter coefficients. This ensures that the filter has useful properties. However, divergence freeness is only respected approximately, the convective operator may need to be modified to preserve skew-symmetry, and there is no coarsening ($\bar{N} = N$).

3.4.2. Spectral cut-off filters

Spectral cut-off filters are divergence-consistent, but only so with respect to the spectral divergence operator $\hat{u}_k \mapsto 2\pi i k^\top \hat{u}_k$ (which acts element-wise in spectral space). For pseudo-spectral discretizations, spectral cut-off filters are therefore natural choices. On our staggered grid however, a spectral cut-off filter would not automatically be such that $\bar{D}\bar{u} = 0$.

3.4.3. Projected filters

By including the projection operator into the filter definition, any filter can be made divergence-consistent [72]. For example, the volume averaging filter Φ^{VA} can be replaced with $\bar{\Phi}^{VA} = \bar{P}\Phi^{VA}$, which is a divergence-consistent filter. However, this makes the filter

non-local. The projection step is also more expensive, which could be a problem when the filter is used to generate many filtered DNS training data samples for a neural closure model.

4. Learning a closure model for the large scale equation

In this section, we present our new closure model formulation: a discrete LES model based on the divergence-consistent face-averaging filter.

4.1. Discrete large eddy simulation

Our "discretize-differentiate-filter" framework has led to equation (11), which describes the exact evolution of the large scale components $\bar{u}$ for a general filter, but still contains the unclosed term $c(u)$ from equation (12). Solving this equation would require access to the underlying DNS solution u. We therefore replace $c(u)$ with a parameterized closure model $m(\bar{u}, \theta) \approx c(u)$, which depends on $\bar{u}$ only [57,60,8]. This produces a new *approximate* large scale velocity field $\bar{v} \approx \bar{u}$. It is defined as the solution to the discrete LES model

$$\mathcal{M}_{\mathrm{DIF}} : \qquad \boxed{\frac{\mathrm{d}\bar{v}}{\mathrm{d}t} = \bar{P}\bar{F}(\bar{v}) + m(\bar{v}, \theta).} \tag{21}$$

Given the discretize-differentiate-filter framework, this constitutes a general LES model formulation, which does not assume yet that the filter is divergence-consistent. We therefore give it the label DIF (divergence-inconsistent formulation).

Since our face-averaging filter is divergence-consistent, we propose an alternative LES model, by replacing c with m in equation (16) instead of in equation (11). The result is a new divergence-consistent LES model:

$$\frac{\mathrm{d}\bar{v}}{\mathrm{d}t} = \bar{P}\left(\bar{F}(\bar{v}) + m(\bar{v}, \theta)\right). \tag{22}$$

This equation is in "pressure-free" form, which was obtained by differentiating the constraint. By reversing the process, i.e. integrating the constraint in time, the model can now be written back into a constrained form in which the pressure reappears:

$$\mathcal{M}_{\mathrm{DCF}} : \qquad \boxed{\begin{aligned} &\bar{D}\bar{v} = 0, \\ &\frac{\mathrm{d}\bar{v}}{\mathrm{d}t} = \bar{F}(\bar{v}) + m(\bar{v}, \theta) - \bar{G}\bar{q}. \end{aligned}} \tag{23}$$

This is our proposed divergence-consistent formulation that will be denoted by "DCF". The derivation of (23) from (22) hinges on the fact that $\bar{D}\bar{v}(0) = \bar{D}\bar{u}(0) = 0$. The pressure field $\bar{q} \approx \bar{p}$ is the filtered pressure field ensuring that the LES solution $\bar{v}$ stays divergence-free. We stress that in our approach no pressure filter needs to be defined explicitly. With a divergence-consistent formulation the filtered pressure can be seen as a Lagrange multiplier.

We note that the system (23) seems to have the same *form* as the LES equations that are common in literature, being obtained by the classic route "filter first, then discretize" (see red route in Fig. 3). One might argue that the divergence-consistent face-averaging filter is just a way to make sure that the operations of differentiation and filtering commute. However, as we mentioned in section 1, there is an important difference: in contrast to the classic approach, in our approach we have precisely defined what the filter is, and the training data ($\bar{u}$) is fully discretization-consistent with our learning target ($\bar{v}$). This is a key ingredient in obtaining model-data consistency and hence stable closure models [19,37].

4.2. Divergence-consistency and LES

We now compare in more detail the properties of the two models, the "divergence-inconsistent" formulation $\mathcal{M}_{\mathrm{DIF}}$ and the "divergence-consistent" formulation $\mathcal{M}_{\mathrm{DCF}}$, see Table 1.

$\mathcal{M}_{\mathrm{DIF}}$ is valid for a general (non-divergence-consistent) filter and leads to a non-divergence free $\bar{v}$. In case a divergence-consistent filter is used, the model $\mathcal{M}_{\mathrm{DIF}}$ is still different from $\mathcal{M}_{\mathrm{DCF}}$, unless $\bar{P}m = m$. This is because one can have an exact commutator error c with the property $\bar{P}c = c$ (meaning $\bar{D}c = 0$) but still learn an approximate commutator error m such that $\bar{P}m \neq m$ (meaning $\bar{D}m \neq 0$).

If the $\mathcal{M}_{\mathrm{DCF}}$ model is used with a volume-averaging filter, inconsistencies appear. For example, while the filtered DNS data is not divergence-free, the DCF model would enforce the LES solution to be (incorrectly) divergence-free.

So far, we have only discussed the two LES formulations in terms of their divergence properties, leaving out other properties that could also be important. In section 5, we compare the four combinations from Table 1 for a turbulent flow test case, and discuss whether they are good choices or not.

4.3. Choosing the objective function

To learn the model parameters θ, we exploit having access to exact filtered DNS data samples $\bar{u}$ and exact commutator errors $c(u)$ obtained through (explicitly) filtered DNS solutions. A straightforward and commonly used approach [33,48,4] is then to minimize a loss function of a-priori type, that only depends on the DNS-solution u. We use the commutator error loss

Table 1
Divergence compatibility chart for LES models (rows) and filter properties (columns). The last row shows our preferred combination.

Model	Filter	$\bar{D}\bar{u} = 0$	$\bar{D}\bar{v} = 0$
$\mathcal{M}_{\text{DIF}}$	VA	False	False
$\mathcal{M}_{\text{DIF}}$	FA	True	True if $\bar{D}m = 0$
$\mathcal{M}_{\text{DCF}}$	VA	False	True
$\mathcal{M}_{\text{DCF}}$	FA	True	True
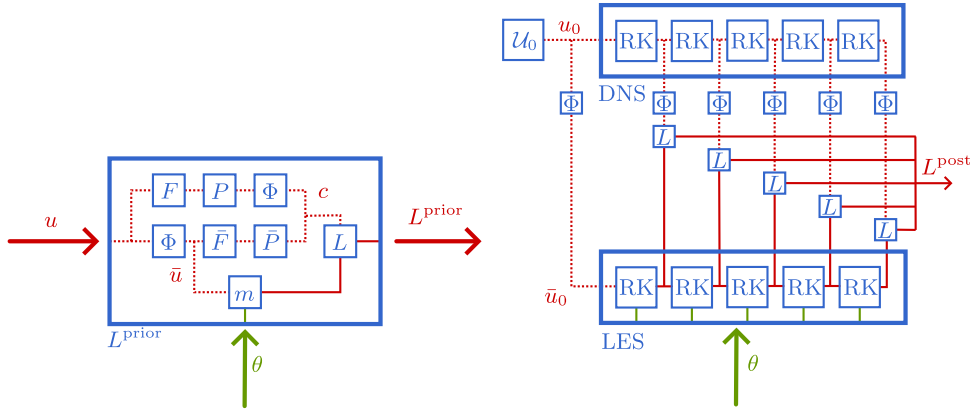
Fig. 6. Computational chain of loss functions. Solid lines are affected by changes in θ. Dotted lines are not affected by θ, and can be precomputed before training. **Left:** A-priori loss function (24). The mean squared error L is computed between the closure model m and the commutator error c. **Right:** A-posteriori loss function (25) (here shown for five unrolled time steps). DNS initial conditions are sampled from the distribution $\mathcal{U}_0$, and filtered (Φ) to produce LES initial conditions. After every time step, the LES solution is compared to the corresponding filtered DNS solution using the mean squared error L. The parameters θ are used in each LES RK time step, but not in the DNS time steps.

$$L^{\text{prior}}(\mathcal{B}, \theta) = \frac{1}{\#\mathcal{B}} \sum_{u \in \mathcal{B}} \frac{\|m(\bar{u}, \theta) - c(u)\|^2}{\|c(u)\|^2} \tag{24}$$

where $\mathcal{B}$ is a batch of $\#\mathcal{B}$ DNS snapshots. Note that (24) does not involve $\bar{v}$, so the effect of the closure model on the LES solution is not taken into account. We therefore call this a-priori training [63]. This approach makes training fast, since only gradients of the neural network itself are required for gradient descent.

Alternatively, one can minimize an a-posteriori loss function, that also depends on the LES solution $\bar{v}$. We use the trajectory loss

$$L^{\text{post}}(u_0, \theta) = \frac{1}{n_{\text{unroll}}} \sum_{i=1}^{n_{\text{unroll}}} \frac{\|\bar{v}_i - \bar{u}_i\|^2}{\|\bar{u}_i\|^2}, \tag{25}$$

where $\bar{u}_i = \Phi u_i$ is obtained by filtering the DNS solution, $u_{i+1} = \text{RK}_{\Delta t}(u_i)$ is obtained using one RK4 time step from section 2.4, u_0 are random initial conditions, and the LES solution $v_{i+1} = \text{RK}_{\Delta t, \mathcal{M}, m, \theta}(v_i)$ is computed using the same time stepping scheme as u_{i+1} but with LES formulation $\mathcal{M}$, closure m, and parameters θ, starting from the exact initial conditions $\bar{v}_0 = \bar{u}_0$.

The parameter n_{unroll} determines how many time steps we unroll. If we choose $n_{\text{unroll}} = 1$, L^{post} will be very similar to L^{prior}. If we choose a large n_{unroll}, we predict long trajectories, and the loss may be more sensitive to small changes in θ ("exploding" gradients). List et al. and Melchers et al. argue that the number of unrolled time steps should depend on the characteristic time scale (Lyapunov time scale) of the problem [43,51]. For chaotic systems (including turbulent flows), L^{post} is expected to grow fast in time, and the number of unrolled time steps should be small. List et al. found good results with $n_{\text{unroll}} \in [30, 60]$ for the incompressible Navier-Stokes equations in 2D [43]. For the chaotic Kuramoto-Sivashinsky equation in 1D, Melchers et al. found $n_{\text{unroll}} = 30$ to give optimal long term results, with $n_{\text{unroll}} = 120$ performing poorly [51]. Kochkov et al. found 32 steps to be ideal for unrolling [35]. For our test case, we choose $n_{\text{unroll}} = 50$, so that the trajectory becomes long enough in time while keeping the number of steps limited. See Appendix B.5 for further details.

The a-priori loss function is easy to evaluate and easy to differentiate with respect to θ, as it does not involve solving the LES ODE given by the model $\mathcal{M} \in \{\mathcal{M}_{\text{DIF}}, \mathcal{M}_{\text{DCF}}\}$. However, minimizing L^{prior} does not take into account the effect of the prediction error on the LES solution error. The a-posteriori loss does take into account this effect, but has a longer computational chain involving the solution of the LES ODE [73,47,43,42,36]. This is illustrated in Fig. 6.

4.4. Choosing the model architecture

Traditionally, closure models are formulated in a continuous setting and they replace the unclosed term $\nabla \cdot (\overline{uu} - \bar{u}\bar{u})$ by either structural or functional models [60]. In recent machine learning approaches, discrete data are inherently used for training the closure model, and the loss function can take into account both structural and functional elements [26]. In this work we use the common approach of using a convolutional neural network for the closure model m [43] (see section 4.4.2), and compare it to a traditional eddy-viscosity model (see section 4.4.1).

4.4.1. Eddy viscosity models

Eddy viscosity models are functional models that consist of adding an additional diffusive term

$$m(\bar{u}, \theta) = \nabla \cdot (2\nu_t \bar{S}) \tag{26}$$

to the continuously filtered Navier-Stokes equations, where $\bar{S} = \frac{1}{2}\left(\nabla \bar{u} + \nabla \bar{u}^\mathsf{T}\right)$ is the large scale strain rate tensor and ν_t is a turbulent viscosity (parameterized by θ). This term models transfer of energy from large to unresolved scales. Note that $\bar{u}(x, t)$ and $m(\bar{u}(\cdot, t), \theta)(x)$ are here continuous quantities that subsequently need to be discretized to $\bar{u}(t)$ and $m(\bar{u}(t), \theta)$.

The Smagorinsky model [69,40] predicts a local viscosity of the form

$$\nu_t = \theta^2 \bar{\Delta}^2 \sqrt{2 \operatorname{tr}(\bar{S}\bar{S})}, \tag{27}$$

where $\bar{\Delta}$ is the filter width and $\theta \in [0, 1]$ is the only model parameter (the Smagorinsky coefficient). In our experiments, this parameter is fitted to filtered DNS data, similar to [64,26]. The model is discretized on the coarse grid, and $\bar{\Delta}$ is taken to be the LES grid size.

4.4.2. Convolutional neural networks (CNNs)

Convolutional neural networks (CNNs) are commonly used in closure models when dealing with structured data [4,64,26,43], and since we are dealing with a structured Cartesian grid, this will be employed here as well. A convolutional node $\mathrm{conv} : (u^n)_{n=1}^{N_{\mathrm{chan}}} \mapsto v$ in a CNN transforms N_{chan} discrete input channel functions into one discrete output function in the following non-linear way:

$$v_I = \sigma\left(b + \sum_{\substack{J \in \{-r, \ldots, r\}^d \\ n \in \{1, \ldots, N_{\mathrm{chan}}\}}} K_J^n u_{I+J}^n\right), \tag{28}$$

where σ is a non-linear activation function, $b \in \mathbb{R}$ is a bias, $K \in \mathbb{R}^{(2r+1)^d \times N_{\mathrm{chan}}}$ is the kernel, and r is the kernel radius.

In a convolutional node it is typically assumed that all the input channels are fields defined in the same grid points. Our closure model, on the other hand, is defined on a staggered grid, with inputs and outputs in the velocity points, whose locations differ in the different Cartesian directions, see Fig. 2. Our CNN closure model is therefore defined as follows:

$$m_{\mathrm{CNN}} = \mathrm{decollocate} \circ \begin{pmatrix} \mathrm{conv} \\ \vdots \\ \mathrm{conv} \end{pmatrix} \circ \ldots \circ \begin{pmatrix} \mathrm{conv} \\ \vdots \\ \mathrm{conv} \end{pmatrix} \circ \mathrm{collocate} \tag{29}$$

where the full vector of degrees of freedom $\bar{u}$ contains d sub-vectors $\bar{u}^\alpha$ used as input channels to the CNN. The degrees of freedom in these sub-vectors belong to their own canonical velocity points. We therefore introduce a so-called collocation function as an initialization layer to the CNN in order to produce quantities that are all defined in the pressure points. The subsequent inner layers map from pressure points to pressure points using kernels of odd diameters. Since the closure term is required in the velocity points, a "decollocation" function is introduced in the last layer to map back from pressure points to velocity points. Here, we use a linear interpolation for both collocation and decollocation functions. It is also possible to use "divergence of a stress tensor" as a decollocation function in order to mimic the structure of the continuous commutator error. However, our commutator error also includes discretization effects, where this form may not be relevant. It would also require more (de)collocation functions to produce the off-diagonal elements of the tensors (which should be the volume corners in 2D and volume edges in 3D).

5. Numerical experiment: forced turbulence in a periodic box

We consider a unit square domain $\Omega = [0, 1]^d$ with periodic boundaries. The DNS initial conditions are sampled from a random velocity field defined through its prescribed energy spectrum $\hat{E}_k$. Similar to [52,61,49], we create an initial energy profile by multiplying a growing polynomial with a decaying exponential as

$$\hat{E}_k = \frac{8\pi}{3\kappa_p^5} \kappa^4 \mathrm{e}^{-2\pi\left(\frac{\kappa}{\kappa_p}\right)^2}, \tag{30}$$

where $k \in \mathbb{Z}^d$ is the wavenumber, $\kappa = \|k\|$, and κ_p is the peak wavenumber. The profile should grow for $\kappa < \kappa_p$, and the decay should take over for $\kappa > \kappa_p$. To further distinguish between the different setups and prevent energy decay during long simulations, we add
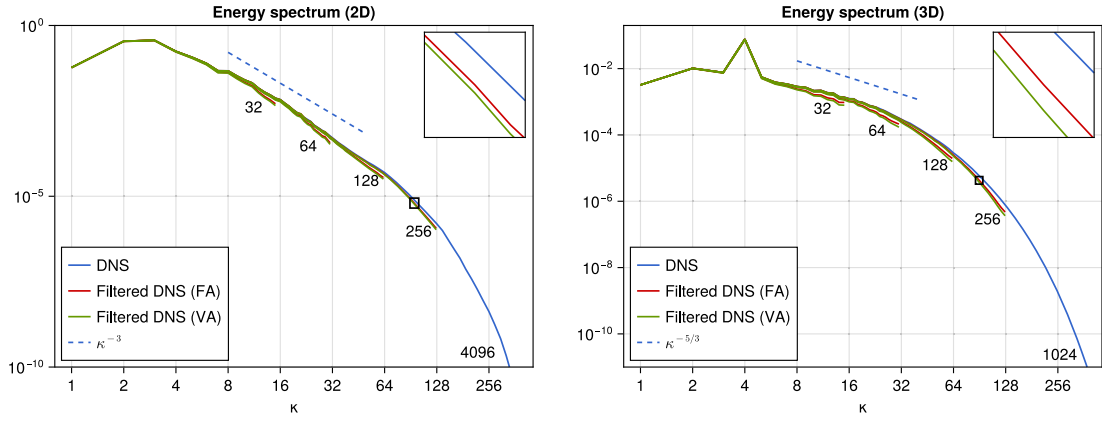
Fig. 7. Kinetic energy spectra of DNS and filtered DNS at final time. The filters are both applied for 4 different filter sizes $\bar{n}$, visible in the four sudden stops at the cut-off wavenumbers $\bar{n}/2$. The numbers below the lines indicate the grid sizes $\bar{n}$ and n. **Left:** 2D simulation, with theoretical scaling κ^{-3}. **Right:** 3D simulation, with theoretical scaling $\kappa^{-5/3}$.

a Kolmogorov-type body force to inject energy into the system, as in [13,35,42]. It is defined as a sinusoidal force at wavenumber $\kappa = 4$ as

$$f^\alpha(x^1, \dots, x^d) = \delta_{\alpha=1} \sin(8\pi x^2). \tag{31}$$

For more details about the initialization procedure, see Appendix B.2.

We perform all simulations in our open source package *IncompressibleNavierStokes.jl*, implemented in the Julia programming language [11]. We use the *KernelAbstractions.jl* [17] framework for implementing back-end agnostic differential operators, *Lux.jl* [54] for neural networks components, *Zygote.jl* [31] for reverse mode automatic differentiation, and *Makie.jl* [18] for visualization. All array operations for DNS, LES, and training are performed on a CUDA-compatible GPU, using *CUDA.jl* [10,9].

5.1. Filtered DNS (2D and 3D)

Before showing results of our new divergence-consistent LES models, we perform a-priori tests to investigate some characteristics of the DNS and filtered DNS solutions. Note that "a-priori" here is used in relation to the analysis of the results (namely before the LES model is employed), while "a-priori" in section 4.3 was related to the training procedure.

We generate two DNS trajectories $u(t)$ (one in 2D, one in 3D), starting from the initial conditions defined above. The 2D simulation is performed with resolution $N = (4096, 4096)$, and the 3D simulation with $N = (1024, 1024, 1024)$. For both simulations, we set $\kappa_p = 20$ and solve until $t_{\text{end}} = 1$ using adaptive time stepping. The Reynolds number is $\text{Re} = 10^4$ in 2D and $\text{Re} = 6000$ in 3D. All array operations are performed on the GPU with double precision floating point numbers in 2D (to show divergence freeness), which works fine for this study even though GPUs are not optimized for double precision, and single precision in 3D (to fit all the arrays in the memory of a single H100 GPU).

For the filter, we consider the face-averaging filter Φ^{FA} and the volume-averaging filter Φ^{VA}. For the 2D setup, we use $\bar{N} = (\bar{n}, \bar{n})$, and for the 3D setup, we use $\bar{N} = (\bar{n}, \bar{n}, \bar{n})$, with $\bar{n} \in \{32, 64, 128, 256\}$.

5.1.1. Energy spectra

Fig. 7 shows the kinetic energy spectra at the final time for the 2D and 3D simulations. The initial velocity field is smooth (containing only low wavenumbers), while the final DNS fields also contain higher wavenumbers. The theoretical slopes of the inertial regions of κ^{-3} in 2D and $\kappa^{-5/3}$ in 3D [57] are also shown (see section Appendix B.1). The inertial region is clearly visible in 2D, but less so in 3D since the DNS-resolution is smaller. The effect of diffusion is visible for $\kappa > 128$ in the 2D plot and for $\kappa > 32$ in the 3D plot, with an attenuation of the kinetic energy. The energy injection wavenumber is visible as a spike in the 3D plot, but not in the 2D plot. The filtered DNS spectra are also shown. A grid of size $(n, \dots, n)$ can only fully resolve wavenumbers in the range $0 \leq \kappa \leq n/2 - 1$, which is visible in the sudden stops of the filtered spectra at $\bar{n}/2 - 1$. The face-averaging filter and the volume-averaging filter have very similar energy profiles. Note that the filtered energy is slightly dampened even before the filter cut-off wavelengths. Since both Φ^{FA} and Φ^{VA} are top-hat like, their transfer functions do not perform sharp cut-offs in spectral space, but affect all wavenumbers [57,8]. The face-averaging filter is damping slightly less than the volume-averaging filter. This is because it averages over one less dimension than the volume-averaging filter, leaving the dimension normal to the face intact. Still, the coarse-graining of the discrete filters creates a spectral cut-off effect that hides the damping of the top-hat transfer functions. This is because the filter width is very close to the coarse-graining spectral cut-off filter width. For further details about the transfer functions, see Appendix E.
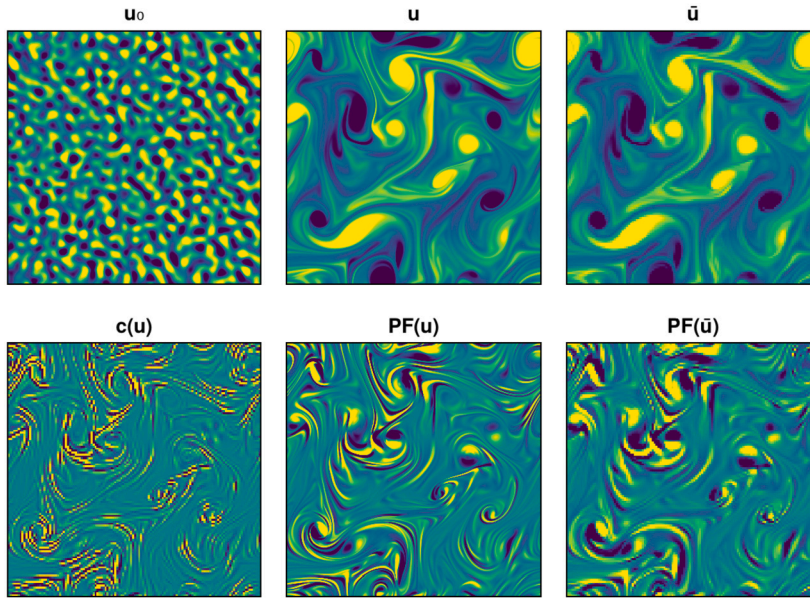
Fig. 8. A-priori results: Discrete curl $-\delta_2 \varphi^1 + \delta_1 \varphi^2$ of various 2D fields $\varphi \in \{u(0), u, \bar{u}, PF(u), \bar{P}\bar{F}(\bar{u}), c(u)\}$. The filter is face-averaging, $\bar{N} = 128^2$.
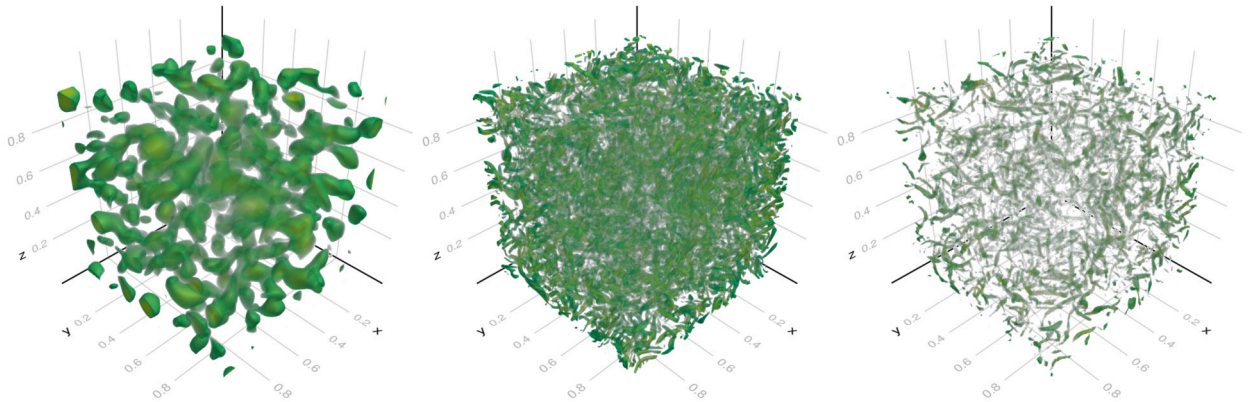

Fig. 9. Vortex cores visualized as 10 isocontours of negative regions of $\lambda_2(S^2 + T^2)$, where λ_2 denotes the second largest eigenvalue, and $S = \frac{1}{2}(\nabla u + \nabla u^\mathsf{T})$ and $T = \frac{1}{2}(\nabla u - \nabla u^\mathsf{T})$ are the symmetric and anti-symmetric parts of the velocity gradient. **Left:** DNS, initial time. **Middle:** DNS, final time. **Right:** Filtered DNS (face-averaging, $\bar{N} = 128^3$), final time.

5.1.2. Filtered fields

Fig. 8 shows the discrete curl $\nabla \times \varphi$ of various 2D fields $\varphi \in \{u(0), u, \bar{u}, PF(u), \bar{P}\bar{F}(\bar{u}), c(u)\}$ for the face-averaging filter with $\bar{N} = 128^2$. We plot the curl since φ is a vector field. Each pixel corresponds to a pressure volume, in which the curl is interpolated for visualization. The filtered field $\bar{u}$ in the top-right corner is clearly unable to represent all the sub-grid fluctuations seen in the DNS field u, but the large eddies of u are still recognizable in $\bar{u}$. We stress again that the aim of our neural closure models is to reproduce $\bar{u}$, without having knowledge of the DNS field u. This will be shown in section 5.2.

Note in particular that the coarse grid right hand side $\bar{P}\bar{F}(\bar{u})$ contains small oscillations which make the discrete curl look grainy. These are due to the under-resolved central-difference discretization on the coarse grid, and are subsequently also present in the discrete commutator error c. The closure model m thus has to predict the oscillations in c in order to correct for those in $\bar{P}\bar{F}(\bar{u})$, using information from the smooth field $\bar{u}$ only. If c was defined by filtering first and then discretizing, as is commonly done in LES, these oscillations would not be part of c and other means of stabilization would be needed to correct for the oscillations in $\bar{P}\bar{F}(\bar{u})$, such as explicit LES [46,21,7] (where the filter is applied to the non-linear convective term in the LES right hand side). This problem has also been addressed in literature; for example, Geurts et al. [25] and Beck and Kurz [6] argue that all commutator errors should be modeled, and Stoffer et al. [70] show that instabilities in the high wavenumbers can occur if the discretization is not included in the commutator error.

Fig. 9 shows the vortex cores of the 3D simulation at initial and final time. The vortex cores are visualized as isocontours of λ_2-criterion [32]. It is defined as negative regions of $\lambda_2(S^2 + T^2)$, where λ_2 denotes the second largest eigenvalue in absolute value

Table 2

Magnitude of various quantities derived from two DNS trajectories $u(t)$ (one in 2D, one in 3D). All quantities are averaged over time. The machine precision for 64-bit numbers is $\epsilon \approx 2.22 \times 10^{-16}$, and for 32-bit numbers $\epsilon \approx 1.19 \times 10^{-7}$.

N	Bits	Filter	$\bar{N}$	$\frac{\|\bar{D}\bar{u}\|}{\|\bar{u}\|}$	$\frac{\|\bar{u}-\bar{P}\bar{u}\|}{\|\bar{u}\|}$	$\frac{\|c-\bar{P}c\|}{\|c\|}$	$\frac{\|c\|}{\|\bar{P}\bar{F}+c\|}$	$\frac{\|\bar{u}\|^2_{\bar{\Omega}}}{\|u\|^2_{\Omega}}$
		FA	32^2	1.5×10^{-14}	2.5×10^{-16}	2.3×10^{-13}	0.56	0.92
		FA	64^2	2.1×10^{-14}	1.9×10^{-16}	3.4×10^{-13}	0.35	0.98
		FA	128^2	3.4×10^{-14}	1.6×10^{-16}	6.1×10^{-13}	0.18	0.99
4096^2	64	FA	256^2	5.3×10^{-14}	1.3×10^{-16}	1.3×10^{-12}	0.077	1.0
		VA	32^2	1.1	0.11	0.11	0.54	0.89
		VA	64^2	0.67	0.0058	0.096	0.33	0.97
		VA	128^2	0.39	0.0019	0.088	0.18	0.99
		VA	256^2	0.19	0.00059	0.12	0.08	1.0
		FA	32^3	2.2×10^{-5}	2.8×10^{-7}	3.9×10^{-6}	0.85	0.63
		FA	64^3	2.2×10^{-5}	1.4×10^{-7}	3.5×10^{-6}	0.70	0.80
		FA	128^3	2.2×10^{-5}	7.6×10^{-8}	4.0×10^{-6}	0.51	0.91
1024^3	32	FA	256^3	2.4×10^{-5}	4.5×10^{-8}	7.6×10^{-6}	0.31	0.96
		VA	32^3	3.3	0.043	0.19	0.80	0.60
		VA	64^3	4.2	0.028	0.15	0.66	0.77
		VA	128^3	4.8	0.017	0.13	0.49	0.89
		VA	256^3	5.0	0.0094	0.13	0.30	0.95

of the 3×3-tensor, and $S = \frac{1}{2}(\nabla u + \nabla u^{\mathsf{T}})$ and $T = \frac{1}{2}(\nabla u - \nabla u^{\mathsf{T}})$ are the symmetric and anti-symmetric parts of the velocity gradient tensor. With the prescribed initial low wavenumber energy spectrum, only large DNS vortex structures are present at the initial time. They are clearly visible in the left plot. As energy gets transferred to higher wavenumbers, smaller turbulent vortex structures are formed (middle plot). The same DNS field still contains larger vortex structures, which become visible after filtering (right plot).

5.1.3. Divergence, commutator errors, and kinetic energy

Table 2 shows the magnitude of various quantities derived from the two DNS trajectories. All quantities $q(u(t))$ are averaged over time with a frequency of $s = 20$ time steps as follows:

$$\langle q \rangle_s = \frac{1}{n_t/s + 1} \sum_{i=0}^{n_t/s} q(u(t_{si})). \tag{32}$$

The considered quantities $q(u)$ are: Normalized divergence $\frac{\|\bar{D}\bar{u}\|}{\|\bar{u}\|}$, magnitude of non-divergence-free part of filtered velocity field $\frac{\|\bar{u}-\bar{P}\bar{u}\|}{\|\bar{u}\|}$, magnitude of non-divergence-free part of commutator error $\frac{\|c-\bar{P}c\|}{\|c\|}$, magnitude of commutator error in the total filtered right hand side $\frac{\|c\|}{\|\bar{P}\bar{F}+\bar{c}\|}$, and resolved kinetic energy $\frac{\|\bar{u}\|^2_{\bar{\Omega}}}{\|u\|^2_{\Omega}}$. The norms are defined as $\|u\| = \sqrt{\sum_{\alpha=1}^{d} \|u^\alpha\|^2}$ for vector fields such as u and $\|u^\alpha\| = \sqrt{\sum_I |u^\alpha_I|^2}$ for scalar fields such as u^α. Additionally, the norm $\| \cdot \|_\Omega$ is weighted by the volume sizes.

It is clear that both $\bar{u}$ and $c(u)$ are divergence-free for the face-averaging filter, in both 2D and 3D. For the volume-averaging filter on the other hand, $\bar{u}$ and $c(u)$ are not divergence-free. At $\bar{N} = 32^2$, the orthogonal projected part $\bar{u} - \bar{P}\bar{u}$ comprises about 1.7% of $\bar{u}$. This is more visible for the commutator error, for which the orthogonal part is 11% of the total commutator error. When the grid is refined to $\bar{N} = 256^2$ however, the non-divergence-free parts of $\bar{u}$ shrink to 0.059% since the flow is more resolved. However, the non-divergence free part of commutator error is 12%, since the commutator error itself is smaller. For volume-averaging with $\bar{N} = 32^3$, the non-divergence free parts of $\bar{u}$ and $c(u)$ are 4.3% and 19% respectively. They shrink to 0.94% and 13% for $\bar{N} = 256^3$. Note that the volume-averaging filter width was chosen to be equal to the grid spacing, but these divergence errors would be larger if we increased the filter width.

For both 2D and 3D, and both filters types, the commutator error becomes smaller when the grid is refined, which is expected since more scales are resolved. The commutator error magnitude does not seem to depend much on whether we use face-averaging or volume-averaging, since they both have the same characteristic filter width. For face-averaging with $\bar{N} = 32^2$ and $\bar{N} = 128^3$, more than half of the total right hand side is due to the commutator error, even though 92% and 91% of the kinetic energy is resolved by $\bar{u}$. For these resolutions, it is very important to have a good closure model. For $\bar{N} = 256^2$, the filtered DNS right hand side is much closer to the corresponding coarse unfiltered DNS right hand side, with $c(u)$ comprising 7.7% of the total right hand side $\bar{P}\bar{F}(\bar{u}) + c(u)$. A closure model is still clearly needed, even though 100% of the energy being resolved by $\bar{u}$. Bae et al. [2] also found the discretization part of the commutator errors to be quite significant, in particular near walls (which we do not consider in this study).

In summary, the DNS and filtered DNS results confirm the theoretical analysis in section 3.3. The volume-averaging filter lacks divergence-consistency of the solution and the closure term. The magnitude of the closure term is similar for both filters. The benefits of divergence-consistency will be demonstrated in the a-posteriori analysis in the subsequent section.
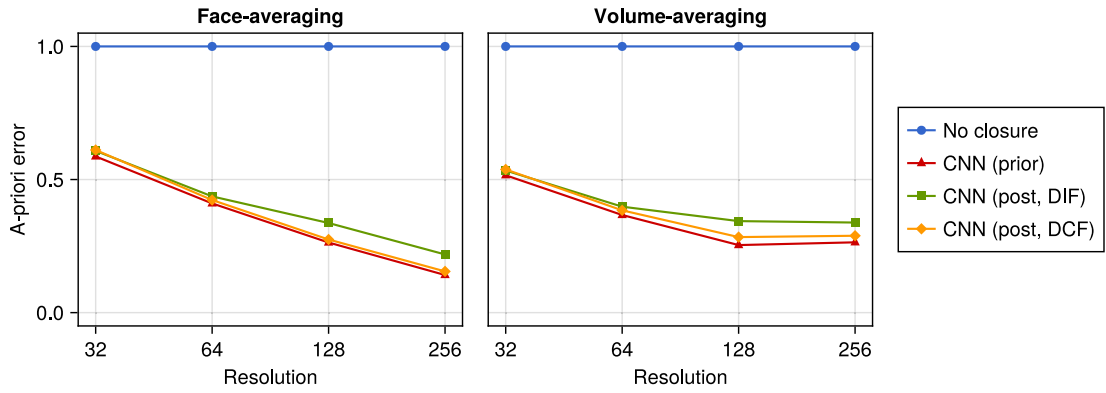
Fig. 10. Relative a-priori errors $\frac{1}{n_u}\sum_u \frac{\|m-c\|}{\|c\|}$ for the testing dataset. **Left:** Face-averaging filter. **Right:** Volume-averaging filter.

5.2. LES (2D)

Next, we turn to the results for the key challenge set out in this paper: testing our neural closure models in an LES setting, with divergence-consistent filters, aiming to approximate the trajectory $\bar{u}(t)$ given $\bar{u}(0)$. We now only consider the 2D problem to reduce the computational time, since the same network is trained repeatedly in multiple configurations. For 3D results, see Appendix G.2. The DNS resolution is set to $N = (4096, 4096)$. The Reynolds number is Re = 6000. The initial peak wavenumber is $\kappa_p = 20$. We use single precision floating point numbers for all computations, including DNS trajectory generation, to reduce memory usage and increase speed. Details about the datasets are found in Appendix B.4.

For the closure model m, we consider three options:

1. No closure model, $m_0 = 0$. This is the baseline model.
2. Smagorinsky closure model, m_S.
3. Convolutional neural closure model, m_{CNN}. The architecture is shown in Appendix B.3.

The no-closure model can be thought of as "coarse DNS", and it is included to show the necessity of a closure model when compared to the filtered DNS reference data. The Smagorinsky model is a traditional closure model. While it is conventionally used in combination with volume-averaging filters, it can also be used with a face-averaging filter. The idea behind the Smagorinsky eddy viscosity $\nu_t = (\theta\bar{\Delta})^2 \sqrt{2\,\mathrm{tr}(\bar{S}\bar{S})}$ is that it is equal to the large-scale strain weighted by the "average sub-filter eddy size" $\theta\bar{\Delta}$, which is parameterized by the fractional constant $\theta \in [0, 1]$. The largest unresolved eddy then has the size $\bar{\Delta}$. This argument also holds for the face-averaging filter, where the largest unresolved eddy has a characteristic size $\bar{\Delta}$, and the averaged unresolved eddy has a size $\theta\bar{\Delta}$ for some coefficient θ (potentially with a different value than for volume-averaging). See Appendix B.5 for details about the training.

5.2.1. A-priori errors

Fig. 10 shows the average relative a-priori errors $\frac{1}{n_u}\sum_u \frac{\|m-c\|}{\|c\|}$ for the testing dataset and the three CNN parameters θ^{prior}, $\theta^{\mathrm{post}}_{\mathrm{DIF}}$, and $\theta^{\mathrm{post}}_{\mathrm{DCF}}$. For the no-closure model m_0, the a-priori error is always $\|0 - c\|/\|c\| = 1$. For both FA and VA and for all grid sizes, the a-priori trained parameters θ^{prior} perform the best. This is expected, since the a-priori loss function used to obtain θ^{prior} is similar to the a-priori error.

5.2.2. A-posteriori errors

The relative a-posteriori error $\frac{1}{n_t}\sum_{i=1}^{n_t}\|\bar{v}_i - \bar{u}_i\|/\|\bar{u}_i\|$ is computed for the trajectory in the testing dataset. A closure model parameter set θ is only used for testing on the same coarse grid and same filter type that it was trained for. Since the turbulent flow is a chaotic system, it does not make sense to plot the point-wise error over long time periods. We therefore plot the errors at time $t = 0.27$. For later time instances, one needs to consider statistical quantities. Fig. 11 shows the average error over time.

For $\mathcal{M}_{\mathrm{DCF}}$ (right), the no-closure and the Smagorinsky closure have similar errors. The CNN is clearly outperforming the other two closures. The a-priori trained CNN is performing significantly better for face-averaging than for volume-averaging. Since the commutator errors are fully consistent with $\mathcal{M}_{\mathrm{DCF}}$ in the face-averaging case, high accuracy can be achieved with a-priori training alone. In the volume-averaging case, the commutator errors are inconsistent with $\mathcal{M}_{\mathrm{DCF}}$, leading to higher errors. When switching to the a-posteriori loss function however, the volume-averaging CNN drastically improves and catches up with the face-averaging one, and the errors become indistinguishable for the two filters. In the face-averaging case, the a-priori trained CNN is already performing well, and training with L^{post} only leads to minor improvements. Note that the a-posteriori training is done starting from the best performing a-priori trained parameters.
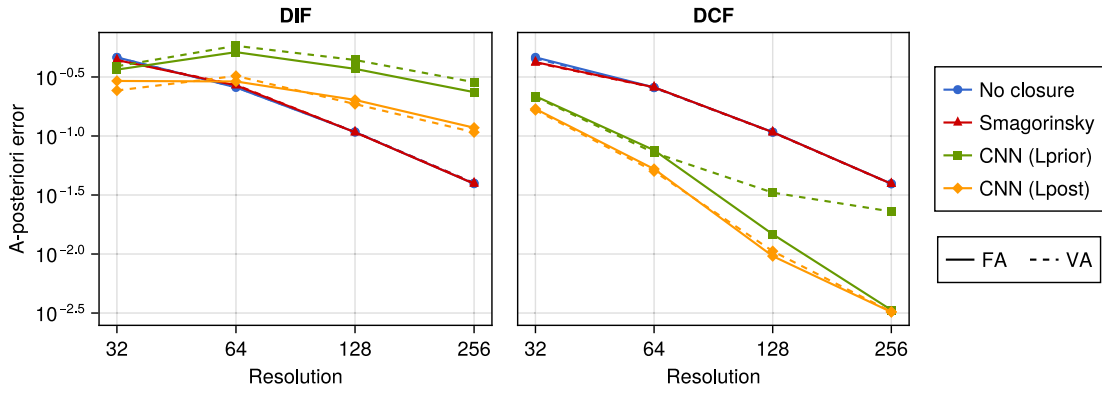
Fig. 11. Relative a-posteriori errors $\frac{1}{n_t}\sum_{i=1}^{n_t}\|\bar{v}_i - \bar{u}_i\|/\|\bar{u}_i\|$ at time $t = 0.27$ for the testing dataset. The CNN is trained separately for each resolution and each filter type with L^{prior} (green squares) and L^{prior}-then-L^{post} (yellow diamonds). **Solid lines:** Face-averaging (Φ^{FA}). **Dashed lines:** Volume-averaging (Φ^{VA}). **Left:** General model $\mathcal{M}_{\text{DIF}}$. **Right:** Divergence-consistent model $\mathcal{M}_{\text{DCF}}$.
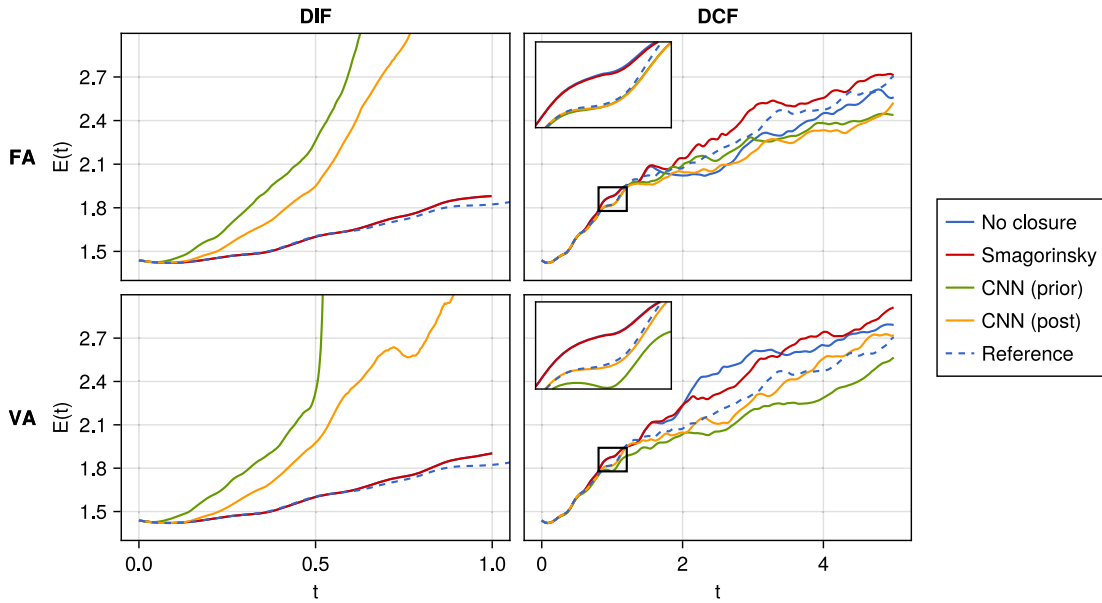

Fig. 12. Total kinetic energy evolution for $\bar{n} = 256$. **Left:** Unprojected closure model $\mathcal{M}_{\text{DIF}}$. **Right:** Constrained model $\mathcal{M}_{\text{DCF}}$. **Top:** Face-averaging filter. **Bottom:** Volume-averaging filter.

For $\mathcal{M}_{\text{DIF}}$ (left), the no-closure and the Smagorinsky closure have similar profiles as for $\mathcal{M}_{\text{DCF}}$. For the no-closure, the two LES formulations are actually identical. For all resolutions except for $\bar{n} = 32$, $m_{\text{CNN}}(\cdot, \theta^{\text{prior}})$ is showing signs of instability and is performing worse than m_0. The a-posteriori training does manage to detect and reduce this error, but it is still unstable and higher than m_0.

5.2.3. Stability

To further investigate the stability, we compute the evolution of the total kinetic energy as a function of time. This is shown in Fig. 12. The model DIF becomes unstable in the long term, so we only compute the energy until the time $t = 1$. The no-closure solution and Smagorinsky solution both stay close to the target energy during the first time unit, after which they become completely decorrelated from the reference solution due to the chaotic nature of turbulence. However, their energy profile does follow the same trend as the reference, staying slightly above. For $\mathcal{M}_{\text{DCF}}$, the a-posteriori trained CNN stays on the reference energy level for much longer than the two other models. As in Fig. 11, $m_{\text{CNN}}(\cdot, \theta^{\text{prior}})$ and $m_{\text{CNN}}(\cdot, \theta^{\text{post}})$ are very similar in the face-averaging case (as can be seen in the zoom-in window), until they reach the point of decorrelation. In the volume-averaging case however, $m_{\text{CNN}}(\cdot, \theta^{\text{prior}})$ performs more poorly than $m_{\text{CNN}}(\cdot, \theta^{\text{post}})$. For $\mathcal{M}_{\text{DIF}}$, the high CNN errors from Fig. 11 are confirmed by the rapid growth of the total kinetic energy. Similar growth in energy of unconstrained neural closure models after a period of seemingly good overlap with the reference energy has been observed by Beck and Kurz [4,37], although in a different configuration. Training with L^{post} improves the stability, and the energy stays close to the reference for a longer period. This was not sufficient to stabilize $\mathcal{M}_{\text{DIF}}$, however, and the energy eventually starts increasing.
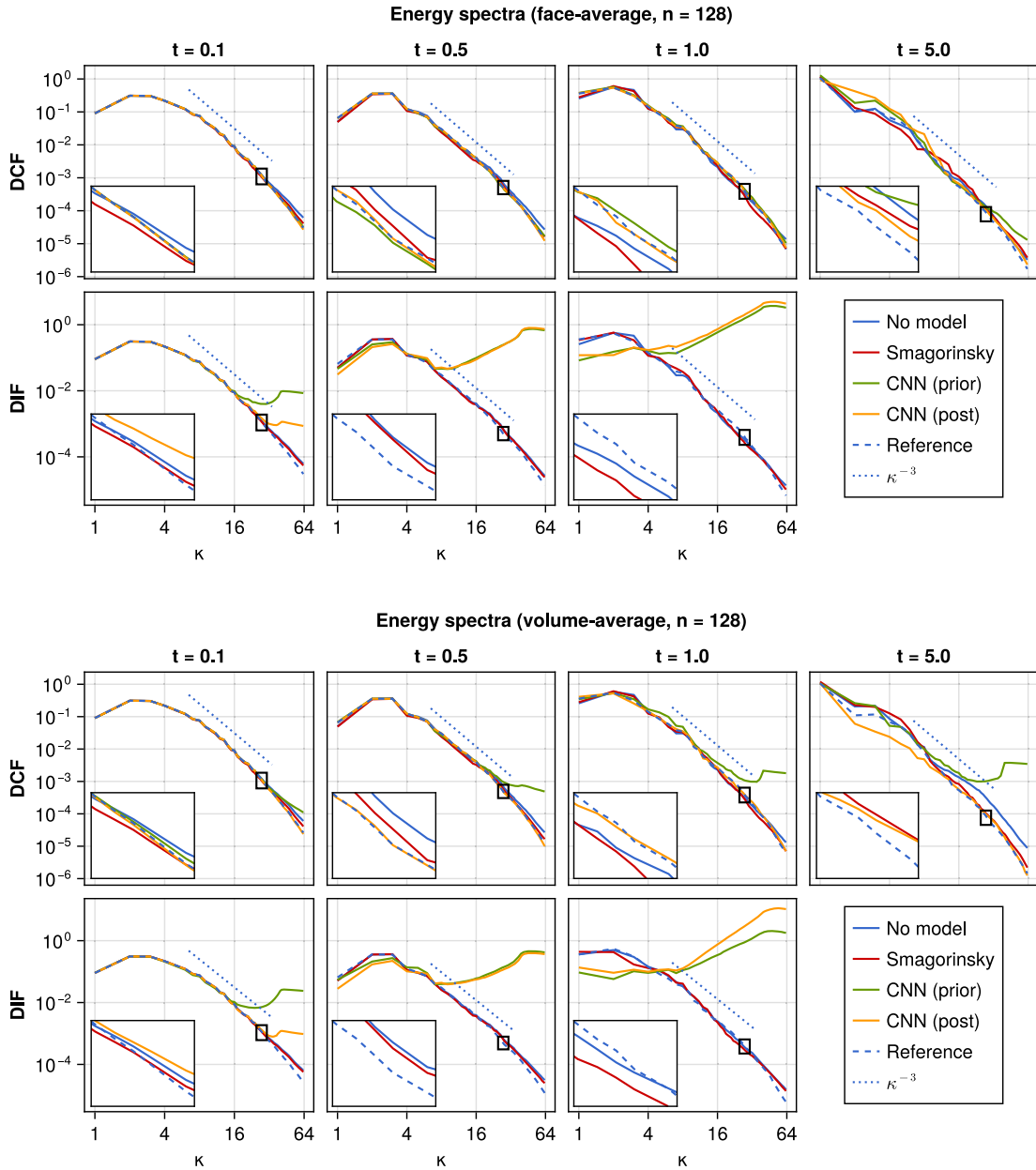
Energy spectra (face-average, n = 128)

Energy spectra (volume-average, n = 128)

Fig. 13. Energy spectra at final time for $\bar{n} = 128$. **Top:** Face-averaging filter. **Bottom:** Volume-averaging filter. Since the $\mathcal{M}_{DIF}$ simulations become unstable, we do not show their spectrum at time $t = 5$.

The growth in kinetic energy and resulting lack of stability for the divergence-inconsistent model can be explained by the energy-conservation properties of our spatial discretization. The convective terms are discretized with a second order central scheme (in so-called divergence form), which can be shown to be equivalent to a skew-symmetric, energy-conserving form *provided that the velocity field is divergence-free* [75]. If the velocity field is not divergence-free, there is no guarantee that the convective terms are energy-conserving, which can lead to growth in kinetic energy and loss of stability.

It is interesting to point out that during the first time unit for $\mathcal{M}_{DCF}$, the CNN models manage to stay close to the target energy level without being explicitly trained to do so. It is also possible to add an energy mismatch term in the loss function, thus encouraging the CNN to produce a correct energy level [43]. Such training could potentially be used to further improve the stability of $\mathcal{M}_{DIF}$.

5.2.4. Energy spectra

The energy spectra at different times are shown in Fig. 13. The no-closure model spectrum is generally close to the reference spectrum, but contains too much energy in the high wavenumbers. This is likely due to the oscillations discussed in the previous
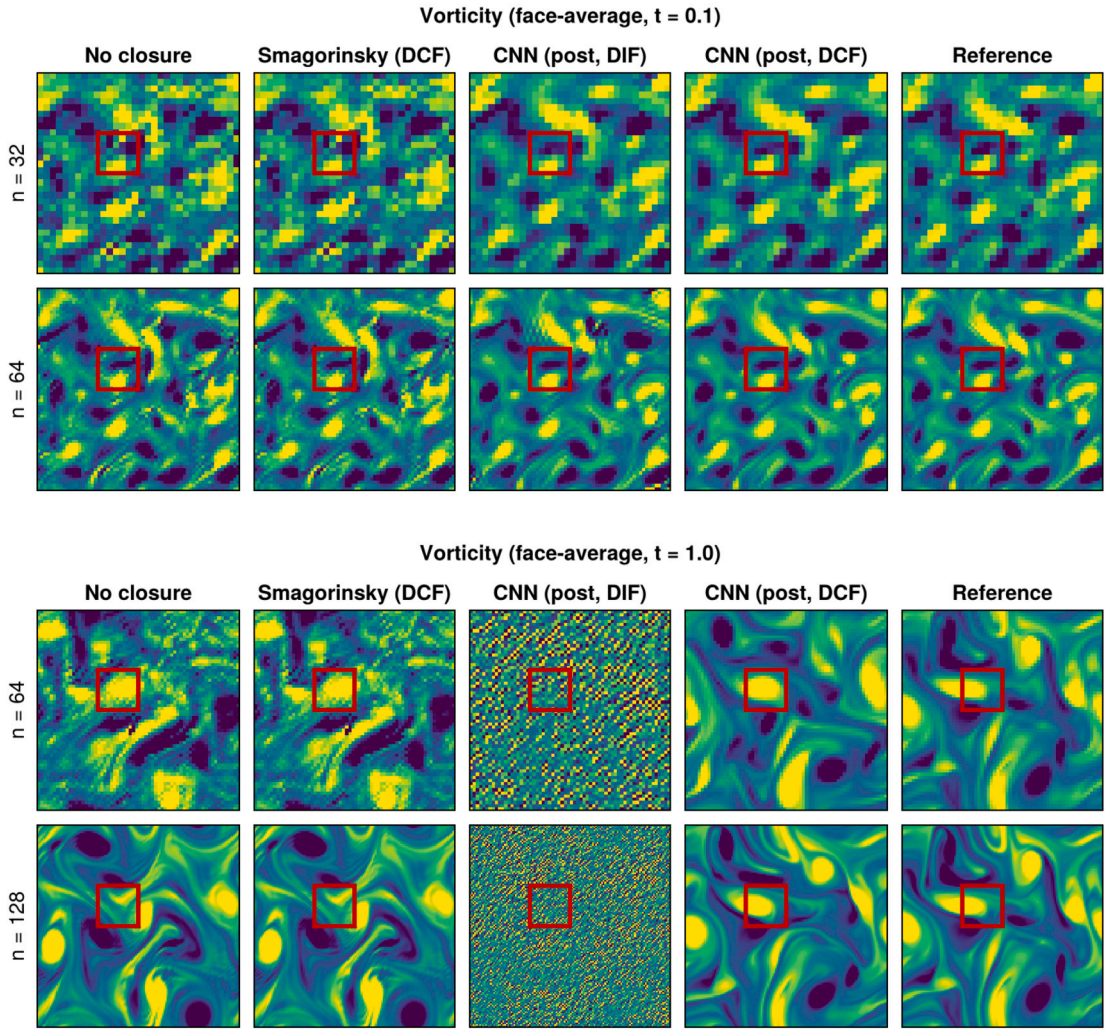
Fig. 14. Vorticity of filtered DNS solution $\bar{u}$ (right) and LES solutions $\bar{v}$ computed using no closure (first column), Smagorinsky model (second column), $\mathcal{M}_{\text{DIF}}$ (third column), and $\mathcal{M}_{\text{DCF}}$ (fourth column) at different times.

sections. The Smagorinsky closure is correcting for this (as intended). For $\mathcal{M}_{\text{DCF}}$, the a-posteriori trained CNN spectrum is visually very close to the reference spectrum, also at the final time, long after the LES trajectory has diverged from the filtered DNS trajectory. This shows that the trained model can capture turbulence statistics correctly even though the system is chaotic. For $\mathcal{M}_{\text{DIF}}$, all the CNNs produce too much energy in the high wave numbers. Training with L^{post} seems to improve this somewhat.

5.2.5. LES solution fields

The curl of the LES solutions (LES vorticity) at the final time is shown in Fig. 14. The reference solution $\bar{u}$ has a smooth vorticity field for all resolutions, since the low-pass filtering operation removes high wavenumber components. The no-closure model solution, on the other hand, shows sharp oscillations. Only the initial conditions are smooth, since $\bar{v}(0) = \bar{u}(0)$. As the solution evolves in time, the central difference scheme used in the right hand side $\bar{P}\bar{F}$ is too coarse for the given Reynolds numbers, and produces well known oscillations [65]. For higher resolutions, these oscillations go away, and $\bar{v}$ starts visually resembling $\bar{u}$.

Adding an $\mathcal{M}_{\text{DCF}}$ constrained CNN closure term trained using $L_{\text{DCF}}^{\text{post}}$ seems to correct for the oscillations of the no-closure model, and we recover the smooth fields with recognizable features from the reference solution (compare $\bar{u}$ and $\bar{v}$ inside the red square). We are effectively doing *large eddy* simulation, as large eddies are visually found in the right positions after simulation. However, if we remove the projection and use the $\mathcal{M}_{\text{DIF}}$ CNN closure model, the solution becomes unstable, even after accounting for this by training with $L_{\text{DIF}}^{\text{post}}$. This confirms the observations from Fig. 11.

We stress again that the CNN closure model is accounting for the total commutator error resulting from the discrete filtering procedure. This commutator error includes the coarse grid discretization error, and also the oscillations produced by the central difference scheme.
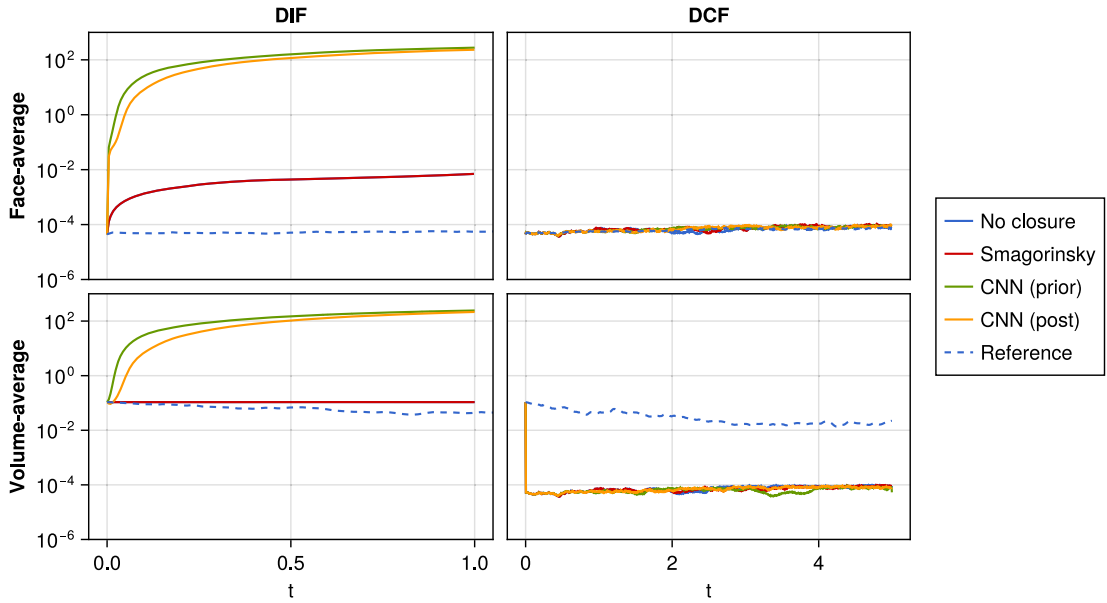
Fig. 15. Divergence evolution for $\bar{n} = 256$. **Left:** Unprojected closure model $\mathcal{M}_{\text{DIF}}$. **Right:** Constrained model $\mathcal{M}_{\text{DCF}}$. **Top:** Face-averaging filter. **Bottom:** Volume-averaging filter.

Table 3
Computational time (in seconds) for DNS and filtering of training, validation, and testing data ($\bar{u}$), a-priori training for 10000 iterations (L^{prior}), and a-posteriori training for 1000 iterations (L^{post}).

$\bar{u}$	Filter	$\bar{n}$	L^{prior}	$L^{\text{post}}_{\text{DIF}}$	$L^{\text{post}}_{\text{DCF}}$
	FA	32	160.6	4460.6	4449.3
	FA	64	244.0	4509.1	4584.7
	FA	128	497.4	4901.2	4853.9
1.5739×10^4	FA	256	1562.3	5822.8	5803.7
	VA	32	157.8	4490.8	4569.4
	VA	64	243.9	4498.5	4608.6
	VA	128	513.2	4914.2	4873.3
	VA	256	1583.6	5793.1	5849.2

5.2.6. Divergence

Fig. 15 shows the evolution of the average divergence $\sqrt{\frac{1}{N} \sum_I (\bar{D}\bar{v})_I^2}$ for the two LES models $\mathcal{M}_{\text{DIF}}$ and $\mathcal{M}_{\text{DCF}}$.

For the face-averaging filter, the filtered DNS divergence (reference) is of the order of 10^{-4} due to the single precision arithmetic. For $\mathcal{M}_{\text{DIF}}$, all closure models produce a divergence increasing in time. This is likely due to the instability observed in the previous figures. For $\mathcal{M}_{\text{DCF}}$, all closure models produce divergences at the same order of magnitude as the reference, since the solution is projected at every time step.

For the volume-averaging filter, the filtered DNS divergence (reference) is of the order of 10^{-1}, since Φ^{VA} does not preserve the divergence constraint. For $\mathcal{M}_{\text{DIF}}$, all closure models produce an increasing divergence, just like for Φ^{FA}. For $\mathcal{M}_{\text{DCF}}$, all closure models produce divergence free solutions, even though the reference solution is actually *not* divergence-free.

5.2.7. Computational cost

The computational time for generating filtered DNS data and training the neural networks is shown in Table 3. One single DNS trajectory is used to compute filtered DNS trajectories for all filters and coarse grid sizes. In total, we compute 8 DNS trajectories. The two models $\mathcal{M}_{\text{DIF}}$ and $\mathcal{M}_{\text{DCF}}$ both produce similar timings, as exactly the same number of operations are performed (but in a different order). The similar times for the three lowest resolutions are likely due to the fact that the same number of time steps are unrolled for all LES resolutions, resulting in an equal number of GPU kernel calls. Since 32^2 and 128^2 are both relatively small, the kernels for the differential operators (convection, Poisson solves, etc.) do not show any speed-up for the lower resolution. This does not seem to be the case for the L^{prior} training however, where most of the kernel calls are to the highly optimized convolutional operators in CUDNN [15].

The benefit of a-posteriori training is clear in situations where the a-priori trained model is unstable (in our case: $\mathcal{M}_{\text{DIF}}$), but for a-priori trained models that are stable and accurate (such as $\mathcal{M}_{\text{DCF}}$ for the face-averaging filter), one could ask whether the additional

cost of a-posteriori training is worth it. We think the additional accuracy is useful, since the learned weights can be reused without having to retrain the model, as long as the same configuration is used (grid size, Reynolds number etc.).

6. Conclusion

The use of neural networks for LES closure models is a promising approach. Neural networks are discrete by nature, and thus require consistent discrete training data. To achieve model-data consistency, we propose the paradigm "discretize, differentiate constraint, filter, and close" (in that particular order). This ensures full model-data consistency and circumvents pressure problems. We do not need to define a pressure filter, only a velocity filter is needed. Our framework allows for training the neural closure model using a-priori loss functions, where the model is trained to predict the target commutator error directly, or using a-posteriori loss functions, where the model is trained to approximate the target filtered DNS-solution trajectory.

To ensure that the filtered DNS velocity stays divergence-free, we employed the divergence-consistent face-averaging filter used by Kochkov et al. [35]. This allows for using a divergence-constrained LES model similar to those commonly used in LES, but with the important difference that the training data obtained through discrete DNS is fully consistent with the LES environment. This resulted in our new divergence-consistent LES model, which was found to be stable with both a-priori and a-posteriori training. Using the same formulation with a divergence-inconsistent filter (VA) did however require a-posteriori training to achieve the same accuracy. A-posteriori training is more expensive than a-priori training, but still has the advantage of increased accuracy.

The divergence-consistent filter stands out from commonly used (volume-averaging) discrete filters, for which the filtered DNS solution is generally not divergence-free. We showed that the resulting LES model can produce instabilities. A-posteriori trained models were found to improve the stability over a-priori trained models, but this was not sufficient to fully stabilize the model in our experiment. With a divergence-consistent formulation, such stability issues did not occur. Another common approach to achieve stability is to add noise to the training data. However, this could destroy the divergence-free property of the data, and appropriate measures would need to be taken to avoid this. We tried adding different levels of noise to the training data, but the resulting validation error was found to be better without noise.

Another important property of our discrete approach is that the (coarse grid) discretization error is included in the training data and learned by the neural network. Turbulence simulations with DNS and LES rely on non-dissipative discretization methods, such as the second order central difference discretization we used in this work, but they produce oscillations and instabilities on coarse grids. This could limit their use in LES, where using coarse grids is one of the main goals. Various smoothing methods, such as explicit LES or (overly) diffusive closure models are commonly used to address this issue. We let the closure model learn to account for the oscillations. The fact that the coarse grid discretization effects (and thus the oscillations) are included in the training data allows the neural network to recognize and correct for these oscillations, even when training with a-priori loss functions.

We realize that the choice of a divergence-consistent face-averaging filter imposes some constraints on the filter choice. The weights have to be uniform (top-hat filter like) to ensure that all the sub-filter velocities cancel out, and the extension to other filters like Gaussians is an open problem. In addition, we took the filter width to be equal to the grid spacing. The face-average filter naturally extends to unstructured grids, as shown in Fig. 2. However, this requires that the DNS grid perfectly overlaps with the faces of the LES grid. This can be achieved by designing the DNS and LES grids at the same time. Lastly, the face-averaging filter is only divergence-consistent with respect to the second-order central difference divergence operator on a staggered grid. We intend to explore the use of filters that are divergence-consistent with respect to higher-order discretization methods (such as the discontinuous Galerkin element method) and non-staggered grids. The "discretize first" approach can also be applied on collocated grids. While is possible to define a face-averaging filter on a collocated *Cartesian* grid, the divergence-consistent property of the filter would no longer hold. On an unstructured collocated grid, it might not make sense to define a face-averaging filter.

Future work will include the use of non-uniform grids with solid walls boundary conditions. This modification will require a change in the neural network architecture, but not in the face-averaging filtering procedure itself. The CNN architecture assumes that the grid is uniform, but weighting the kernel stencil with non-uniform volume sizes could be a way to design a discretization-informed neural network. We intend to exploit the fact that discrete DNS boundary conditions naturally extend to the LES model for the face-average filter. We also intend to incorporate other constraints into the LES model, such as conservation of quantities of interest, in particular kinetic energy conservation, as studied in [74].

CRediT authorship contribution statement

Syver Døving Agdestein: Writing – original draft, Visualization, Validation, Software, Methodology, Conceptualization. **Benjamin Sanderse:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Formal analysis.

Software and reproducibility statement

The Julia scripts and source code used to generate all the results are available at
https://github.com/agdestein/IncompressibleNavierStokes.jl.
It is released under the MIT license. Each simulation was run on a single Nvidia H100 GPU on the Dutch national Snellius supercomputer.
To produce the results of this article, pseudo-random numbers were used for

- data generation (DNS initial conditions for training, validation, and testing data);
- neural network parameter initialization;
- batch selection during stochastic gradient descent.

The scripts include the seeding numbers used to initialize the pseudo-random number generators. Given a seeded pseudo-random generator, the code is deterministic and the results are reproducible.

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

During the preparation of this work the authors used GitHub Copilot in order to propose wordings and mathematical typesetting. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

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

Appendix A. Finite volume discretization

In this work, the integral form of the Navier-Stokes equations is considered, which is used as starting point to develop a spatial discretization:

$$\frac{1}{|\mathcal{O}|} \int_{\partial \mathcal{O}} u \cdot n \, \mathrm{d}\Gamma = 0, \tag{A.1}$$

$$\frac{\mathrm{d}}{\mathrm{d}t} \frac{1}{|\mathcal{O}|} \int_{\mathcal{O}} u \, \mathrm{d}\Omega = \frac{1}{|\mathcal{O}|} \int_{\partial \mathcal{O}} \left(-u u^{\mathsf{T}} - pI + \nu \nabla u \right) \cdot n \, \mathrm{d}\Gamma + \frac{1}{|\mathcal{O}|} \int_{\mathcal{O}} f \, \mathrm{d}\Omega, \tag{A.2}$$

where $\mathcal{O} \subset \Omega$ is an arbitrary control volume with boundary $\partial\mathcal{O}$, normal n, surface element $\mathrm{d}\Gamma$, and volume size $|\mathcal{O}|$. We have divided by the control volume sizes in the integral form, so that system (A.1)-(A.2) has the same dimensions as the system (1)-(2).

A.1. Staggered grid configuration

In this section we describe a finite volume discretization of equations (A.1)-(A.2). Before doing so, we introduce our notation, which is such that the mathematical description of the discretization closely matches the software implementation.

The d spatial dimensions are indexed by $\alpha \in \{1, \dots, d\}$. The α-th unit vector is denoted $2h_\alpha = (2h_{\alpha\beta})_{\beta=1}^d$, where the (half) Kronecker symbol $h_{\alpha\beta}$ is $1/2$ if $\alpha = \beta$ and 0 otherwise. The Cartesian index $I = (I_1, \dots, I_d)$ is used to avoid repeating terms and equations d times, where I_α is a scalar index (typically one of i, j, and k in common notation). This notation is dimension-agnostic, since we can write u_I instead of u_{ij} in 2D or u_{ijk} in 3D. In our Julia implementation of the solver we use the same Cartesian notation (`u[I]` instead of `u[i, j]` or `u[i, j, k]`).

For the discretization scheme, we use a staggered Cartesian grid as proposed by Harlow and Welch [29]. Staggered grids have excellent conservation properties [41,56], and in particular their exact divergence-freeness is important for this work. Consider a rectangular domain $\Omega = \prod_{\alpha=1}^d [a_\alpha, b_\alpha]$, where $a_\alpha < b_\alpha$ are the domain boundaries and $\prod$ is a Cartesian product. Let $\Omega = \bigcup_{I \in \mathcal{I}} \Omega_I$ be a partitioning of Ω, where $\mathcal{I} = \prod_{\alpha=1}^d \{\frac{1}{2}, 2 - \frac{1}{2}, \dots, N_\alpha - \frac{1}{2}\}$ are volume center indices, $N = (N_1, \dots, N_d) \in \mathbb{N}^d$ are the number of volumes in each dimension, $\Omega_I = \prod_{\alpha=1}^d \Delta_{I_\alpha}^\alpha$ is a finite volume, $\Gamma_I^\alpha = \Omega_{I-h_\alpha} \cap \Omega_{I+h_\alpha} = \prod_{\beta \neq \alpha} \Delta_{I_\beta}^\beta$ is a volume face, $\Delta_i^\alpha = \left[x_{i-\frac{1}{2}}^\alpha, x_{i+\frac{1}{2}}^\alpha \right]$ is a volume edge, $x_0^\alpha, \dots, x_{N_\alpha}^\alpha$ are volume boundary coordinates, and $x_i^\alpha = \frac{1}{2} \left(x_{i-\frac{1}{2}}^\alpha + x_{i+\frac{1}{2}}^\alpha \right)$ for $i \in \{1/2, \dots, N_\alpha - 1/2\}$ are volume center coordinates. We also define the operator δ_α which maps a discrete scalar field $\varphi = (\varphi_I)_I$ to

$$(\delta_\alpha \varphi)_I = \frac{\varphi_{I+h_\alpha} - \varphi_{I-h_\alpha}}{|\Delta_{I_\alpha}^\alpha|}. \tag{A.3}$$
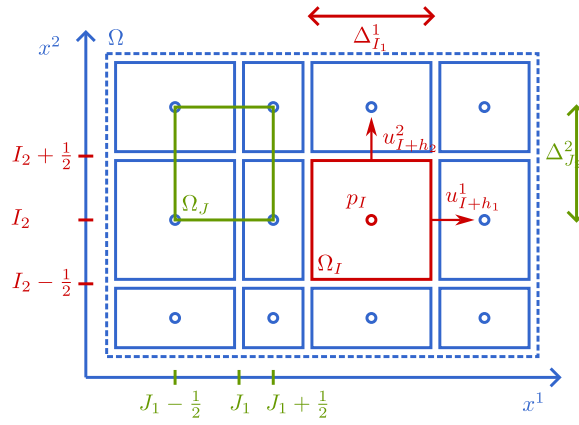
Fig. A.16. Finite volume discretization on a staggered grid. Note that the grid can be non-uniform, as long as each volume in a given column has the same width and each volume in a given row has the same height. Here, I and J are two arbitrary Cartesian indices, with $I \in \mathcal{I}$ in a volume center and $J \in \mathcal{I} + h_1 + h_2$ in a volume corner for illustrative purposes.

It can be interpreted as a discrete equivalent of the continuous operator $\frac{\partial}{\partial x^\alpha}$. All the above definitions are extended to be valid in volume centers $I \in \mathcal{I}$, volume faces $I \in \mathcal{I} + h_\alpha$, or volume corners $I \in \mathcal{I} + \sum_{\alpha=1}^{d} h_\alpha$. The discretization is illustrated in Fig. A.16.

A.2. Equations for unknowns

We now define the unknown degrees of freedom. The average pressure in Ω_I, $I \in \mathcal{I}$ is approximated by the quantity $p_I(t)$. The average α-velocity on the face Γ_I^α, $I \in \mathcal{I} + h_\alpha$ is approximated by the quantity $u_I^\alpha(t)$. Note how the pressure p and the d velocity fields u^α are each defined in their own canonical positions x_I and x_{I+h_α} for $I \in \mathcal{I}$. This is illustrated for a given volume I in Fig. A.16. In the following, we derive equations for these unknowns.

Using the pressure control volume $\mathcal{O} = \Omega_I$ with $I \in \mathcal{I}$ in the integral constraint (A.1) and approximating the face integrals with the mid-point quadrature rule $\int_{\Gamma_I} u \, d\Gamma \approx |\Gamma_I| u_I$ results in the discrete divergence-free constraint

$$\sum_{\alpha=1}^{d} (\delta_\alpha u^\alpha)_I = 0. \tag{A.4}$$

Note how dividing by the volume size results in a discrete equation resembling the continuous one (since $|\Omega_I| = |\Gamma_I^\alpha| |\Delta_{I_\alpha}^\alpha|$).

Similarly, choosing an α-velocity control volume $\mathcal{O} = \Omega_I$ with $I \in \mathcal{I} + h_\alpha$ in equation (A.2), approximating the volume- and face integrals using the mid-point quadrature rule, and replacing remaining spatial derivatives in the diffusive term with a finite difference approximation gives the discrete momentum equations

$$\frac{\mathrm{d}}{\mathrm{d}t} u_I^\alpha = - \sum_{\beta=1}^{d} (\delta_\beta(u^\alpha u^\beta))_I + \nu \sum_{\beta=1}^{d} (\delta_\beta \delta_\beta u^\alpha)_I + f^\alpha(x_I) - (\delta_\alpha p)_I, \tag{A.5}$$

where we made the assumption that f is constant in time for simplicity. The outer discrete derivative in $(\delta_\beta \delta_\beta u^\alpha)_I$ is required at the position I, which means that the inner derivative is evaluated as $(\delta_\beta u^\alpha)_{I+h_\beta}$ and $(\delta_\beta u^\alpha)_{I-h_\beta}$, thus requiring $u_{I-2h_\beta}^\alpha$, u_I^α, and $u_{I+2h_\beta}^\alpha$, which are all in their canonical positions. The two velocity components in the convective term $u^\alpha u^\beta$ are required at the positions $I - h_\beta$ and $I + h_\beta$, which are outside the canonical positions. Their value at the required position is obtained using averaging with weights $1/2$ for the α-component and with linear interpolation for the β-component. This preserves the skew-symmetry of the convection operator, such that energy is conserved (in the convective term) [75].

Appendix B. Numerical experiment details

B.1. Energy spectra

For our discretization, we define the energy at a wavenumber k as $\hat{E}_k = \frac{1}{2}\|\hat{u}_k\|^2$, where $\hat{u}^\alpha = \mathrm{DFT}(u^\alpha)$ is the discrete Fourier transform of u. Since $k \in \mathbb{Z}^d$, it is not immediately clear how to compute a discrete equivalent of the scalar energy spectrum as a function of $\|k\|$. We proceed as follows. The energy at a scalar level $\kappa > 0$ is defined as the sum over all energy components of the dyadic bin $\mathcal{K}_\kappa = \{k \mid \kappa/a \leq \|k\| \leq \kappa a\}$ as

$$\hat{E}_\kappa = \sum_{k \in \mathcal{K}_\kappa} \hat{E}_k. \tag{B.1}$$

Table B.4

CNN architecture, with $\epsilon = 1/100$. The total radius is 8, which means that the component $m(\bar{u}, \theta)_I$ depends on the components $\bar{u}_{I+J}$ for $J \in [-8, 8]^2$. In comparison, the diffusion operator has a radius of 1.

Layer	Radius	Channels	Activation	Bias	Parameters
Interpolate$_{u \to p}$					
Conv	2	$2 \to 24$	tanh	Yes	1224
Conv	2	$24 \to 24$	tanh	Yes	14424
Conv	2	$24 \to 24$	tanh	Yes	14424
Conv	2	$24 \to 24$	tanh	Yes	14424
Conv	2	$24 \to 2$	$x \mapsto x$	No	1200
Interpolate$_{p \to u}$					
					45696

The parameter $a > 1$ determines the width of the interval. Lumley argues to use the golden ratio $a = (1 + \sqrt{5})/2 \approx 1.6$ [23]. Note that there is no averaging factor in front of the sum (B.1), even though the number of wavenumbers in the set increases with κ.

For homogeneous decaying isotropic turbulence, the spectrum should behave as follows. In the inertial region, for large Reynolds numbers, the theoretical decay of $\hat{E}$ should be $\hat{E}_\kappa = \mathcal{O}(\kappa^{-3})$ in 2D and $\mathcal{O}(\kappa^{-5/3})$ in 3D [57]. For the lowest wavenumbers, we should have $\hat{E}_\kappa = \mathcal{O}(\kappa^4)$ or $\hat{E}_\kappa = \mathcal{O}(\kappa^2)$.

It is also common to use a linear bin such as $\mathcal{K}_\kappa = \{k \mid \kappa - \frac{1}{2} \leq \|k\| < \kappa + \frac{1}{2}\}$ [52,61,49]. However, this leads to a different power law scaling in the inertial range than the well known κ^{-3} and $\kappa^{-5/3}$.

B.2. Initial conditions

To generate initial conditions in a periodic box, we consider a prescribed energy spectrum $\hat{E}_k$. We want to create an initial velocity field u with the following properties:

- The Fourier transform of u, noted $\hat{u}$, should be such that $\frac{1}{2}\|\hat{u}_k\|^2 = \hat{E}_k$ for all k.
- u should be divergence-free with respect to our discretization: $Du = 0$.
- u should be parameterized by controllable random numbers, such that a wide variety of initial conditions can be generated.

These properties are achieved by sampling a velocity field in spectral space, projecting (making it divergence-free), transforming to physical space, and projecting again. In detail, let $a_k = \sqrt{2\hat{E}_k} e^{2\pi i \tau_k}$, where $\tau_k = \sum_{\alpha=1}^d \xi_k^\alpha$ is phase shift, $\xi_k^\alpha \sim \mathcal{U}[0, 1]$ is a random uniform number if $k_\beta \geq 0$ for all β. If $k_\beta < 0$ for any β, we add the symmetry constraint $\xi_k^\alpha = \text{sign}(k_\alpha)\xi_{|k|}^\alpha$ where $|k| = (|k_\beta|)_{\beta=1}^d$. Then $\|\hat{u}_k\| = |a_k|$, and a_k has a random phase shift. We then multiply the scalar a_k with a random unit vector e_k projected onto the divergence-free spectral grid as follows: $\hat{u}_k^\alpha = a_k \hat{P}_k e_k^\alpha / \|\hat{P}_k e_k^\alpha\|$, where $\hat{P}_k = I - \frac{kk^\top}{k^\top k} \in \mathbb{C}^{d \times d}$ is a projector for each k ensuring that $2\pi i k^\top \hat{u}_k = 0$ for all k (which is the equivalent of $\nabla \cdot u = 0$ in spectral space) [57]. The normalization with respect to $\|\hat{P}_k e_k^\alpha\|$ ensures that no energy is lost in the projection step. In 2D, we choose a random vector on the unit circle $e_k = (\cos(\theta_k), \sin(\theta_k))$ with $\theta_k \sim \mathcal{U}[0, 2\pi]$. In 3D, we choose a random vector on the unit sphere $e_k = (\sin(\theta_k)\cos(\phi_k), \sin(\theta_k)\sin(\phi_k), \cos(\phi_k))$ with $\theta_k \sim \mathcal{U}[0, \pi]$ and $\phi_k \sim \mathcal{U}[0, 2\pi]$. Finally, we obtain the velocity field u by taking the inverse discrete Fourier transform, and also projecting it again since divergence-freeness on the "spectral grid" and on the staggered grid are slightly different. This gives the random initial field

$$u = P \, \text{DFT}^{-1}(\hat{u}). \tag{B.2}$$

Note that the second projection may result in a slight loss of energy, but since u is already divergence-free on the spectral grid, our experience is that the loss is non-significant.

B.3. CNN architecture

The CNN architecture is shown in Table B.4. We use periodic padding. For the last convolutional layer, we use no activation and no bias, in order not to limit the expressiveness of m. For the inner layers, we use bias and the tanh activation function.

The choice of channel sizes is chosen solely to have sufficient expressive capacity in the closure model. The kernel radius on the other hand, is chosen to be small ($r = 2$, diameter 5) to ensure that the closure model uses local information. In addition, the resulting small stencils that are learned can possibly be interpreted as discrete differential operators of finite difference type, separated by simple non-linearities. In this way, the CNN can be thought of as a generalized Taylor series expansion of the commutator error in terms of the filtered velocity field, similar to certain continuous filter expansions [60]. We do not investigate this further in this study, but it could be a direction for future research.

The same CNN architecture m is used for all grids and filters. We choose a simple architecture since the goal of the study is not to get the most accurate closure model, but rather to compare different filters, LES formulations, and loss functions for the same closure architecture. For the LES formulation, we only consider the two models $\mathcal{M}_{\text{DIF}}$ and $\mathcal{M}_{\text{DCF}}$.

Table B.5

Optimized Smagorinsky coefficients for each coarse resolution, filter type, and LES model.

$\bar{n}$	$\theta_{\mathrm{DIF}}^{\mathrm{FA}}$	$\theta_{\mathrm{DIF}}^{\mathrm{VA}}$	$\theta_{\mathrm{DCF}}^{\mathrm{FA}}$	$\theta_{\mathrm{DCF}}^{\mathrm{VA}}$
32	0.129	0.131	0.144	0.142
64	0.114	0.116	0.143	0.141
128	0.061	0.064	0.096	0.093
256	0.000	0.000	0.023	0.014
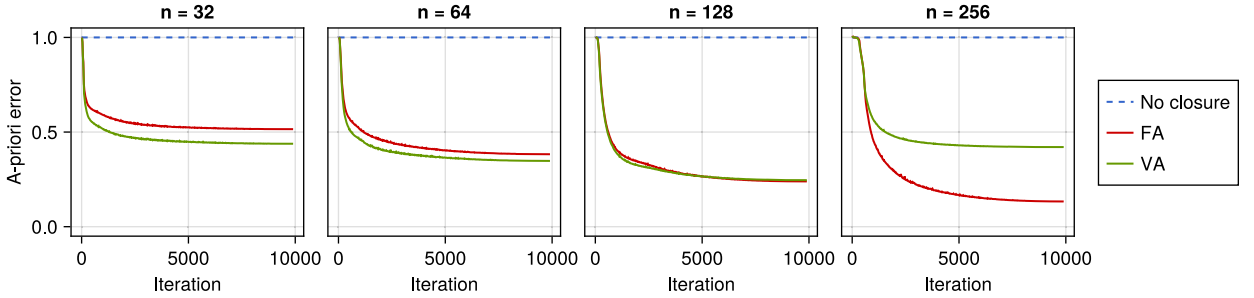
Fig. B.17. Relative a-priori error on validation set during a-priori training for 10^4 iterations. From left to right: $\bar{n} = 32, 64, 128, 256$.

B.4. Data generation

To create data, we run 8 DNS simulations of size $N = 4096^2$. We use adaptive time-stepping. For every random initial flow field $u(0)$, we let the DNS run for a burn-in time $t_{\mathrm{burn}} = 0.5$ to initialize the flow beyond the artificial initial spectrum (30). We then start saving $\bar{u}$ and c every time step until $t_{\mathrm{end}} = 5$. Every 50 time steps we compute $\bar{u}$ and $c(u)$ on four coarse grids of size $\bar{n} \in \{32, 64, 128, 256\}$. The first 6 trajectories are used for training, and the remaining 2 for validation and testing.

B.5. Training

Both the Smagorinsky model and the CNN are parameterized and require training. Since the Smagorinsky parameter is a scalar, we perform a grid search to find the optimal parameter for each of the grid sizes, filter types and projection orders. The relative a-posteriori error for the training set is evaluated. We choose the value of $\theta \in \{0, 1/1000, 2/1000, \dots, 300/1000\}$ that gives the lowest training error. The resulting Smagorinsky constants are shown in Table B.5. Note that the Smagorinsky constants are theoretically grid-independent, since information about the grid is incorporated using the filter width that enters the expression separately. We still optimize the coefficient for each grid size in order to achieve a fair comparison with the CNN. It is important to stress that the Smagorinsky closure term (divergence of weighted strain tensor in equation (26)) is *not* divergence-free, just like the other right hand side terms like convection and diffusion. As a result, the optimal Smagorinsky coefficients for $\mathcal{M}_{\mathrm{DIF}}$ at the higher LES resolutions (where the CNN is unstable) go to zero, thus creating a divergence-free right hand side.

For the CNN, the initial model parameters θ_0 are sampled from a uniform distribution. They are improved by minimizing the stochastic loss function using the ADAM optimizer [34]. The gradients are obtained using reverse mode automatic differentiation.

We start by training using the a-priori loss function (24). We learn one set of parameters θ^{prior} for each of the training grids and filter types. Each time, the model is trained for 50 epochs using the Adam optimizer [34] with default weight decay and momentum parameters. Each epoch consists of iterating through the 198 training batches, each containing 64 $(\bar{u}, c)$ snapshot pairs, resulting in roughly 10^4 iterations of stochastic gradient descent. The learning rate is set before each epoch using a cosine annealing scheduler [44] with initial learning rate 10^{-3} and final learning rate 10^{-6}. Every 20 iterations the a-priori error is evaluated on the validation dataset. The validation error is shown in Fig. B.17. Note that the commutator error is different for each LES resolution and filter type, and so the relative a-priori error is not directly comparable between setups. The parameters giving the lowest validation error are retained after training.

Since the a-priori loss L^{prior} does not take into account the effect of the LES model we use, we fine-tune the a-priori trained CNN parameters by training using the a-posteriori loss function $L_{\mathcal{M}}^{\mathrm{post}}$. This time, the LES model $\mathcal{M} \in \{\mathcal{M}_{\mathrm{DIF}}, \mathcal{M}_{\mathrm{DCF}}\}$ is part of the loss function definition. We use $n_{\mathrm{unroll}} = 50$, and train for 1000 iterations, as shown in Fig. B.18. Since the weights are already converged from the a-priori training, we use a smaller learning rate with the scheduler, starting at 10^{-4} and ending at 10^{-6}. The best parameters from the a-priori training (θ^{prior}) are used as initial parameters for the a-posteriori training. The Adam optimizer is reinitialized without the history terms from the a-priori training session. The parameters $\theta_{\mathcal{M}}^{\mathrm{post}}$ giving the lowest a-posteriori error on the validation dataset during training are retained after training.

For FA with $\bar{n} = 256$, the parameters found with a-priori training are already close to the optimum, and subsequent a-posteriori training only leads to minor improvements. For VA with $\bar{n} = 256$, the same a-priori training validation error is much higher than for FA. A-posteriori training therefore improves the validation error significantly, and eventually reaches the same error as for FA. This
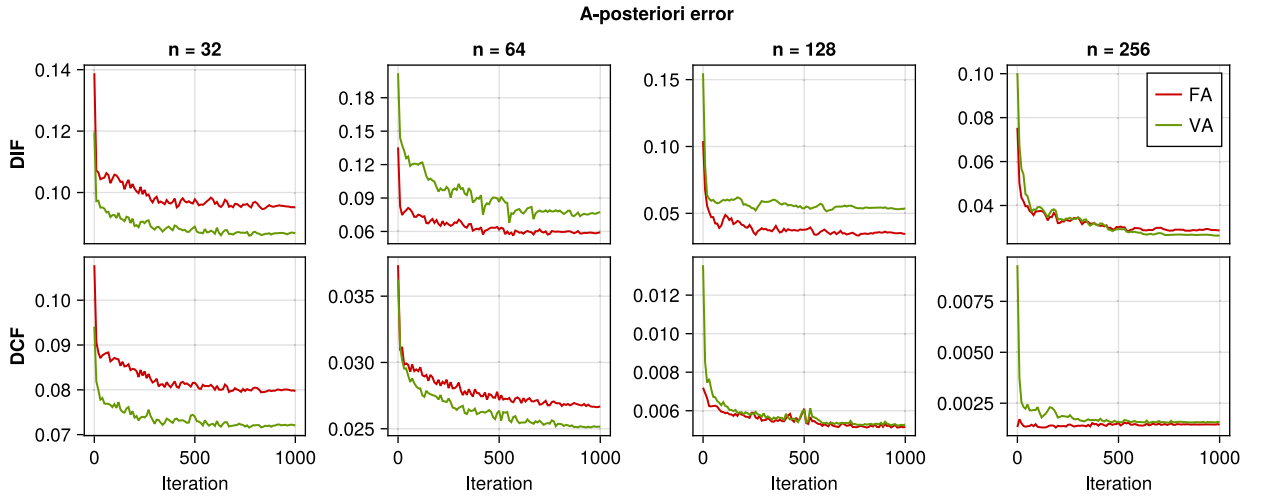
A-posteriori error

Fig. B.18. Relative a-posteriori error on validation set during a-posteriori training for 1000 iterations. From left to right: $\bar{n} = 32, 64, 128, 256$.

is likely due to the fact that for VA, the commutator error used in a-priori training is inconsistent with $\mathcal{M}_{\mathrm{DCF}}$. Training with $L_{\mathrm{DCF}}^{\mathrm{post}}$ corrects for this inconsistency. For FA, the computed commutator errors are fully consistent with $\mathcal{M}_{\mathrm{DCF}}$, and a-posteriori training is not needed.

Appendix C. Divergence-free filter

In this appendix we give the proof that the face-averaging filter is divergence-free. The proof is a natural consequence of the continuous divergence-theorem.

C.1. Simple example

Consider a uniform 2D-grid with 3×3 fine volumes in each coarse volume. The divergence-free constraint for a volume $\Omega_{i,j}$ of the fine grid reads:

$$(Du)_{i,j} = (\delta_1 u^1)_{i,j} + (\delta_2 u^2)_{i,j} = 0, \tag{C.1}$$

where

$$(\delta_1 u^1)_{i,j} = \frac{u^1_{i+\frac{1}{2},j} - u^1_{i-\frac{1}{2},j}}{\Delta^1} \quad \text{and} \quad (\delta_2 u^2)_{i,j} = \frac{u^2_{i,j+\frac{1}{2}} - u^2_{i,j-\frac{1}{2}}}{\Delta^2}. \tag{C.2}$$

Let (a, b) and (i, j) be coarse-grid and fine-grid indices such that $\Omega_{i,j}$ is in the center of $\bar{\Omega}_{a,b}$. The four face-averaged velocities at the boundary of $\bar{\Omega}_{a,b}$ are

$$\begin{aligned}
\bar{u}^1_{a+\frac{1}{2},b} &= \frac{1}{3} u^1_{i+\frac{3}{2},j-1} + \frac{1}{3} u^1_{i+\frac{3}{2},j} + \frac{1}{3} u^1_{i+\frac{3}{2},j+1} \\
\bar{u}^1_{a-\frac{1}{2},b} &= \frac{1}{3} u^1_{i-\frac{3}{2},j-1} + \frac{1}{3} u^1_{i-\frac{3}{2},j} + \frac{1}{3} u^1_{i-\frac{3}{2},j+1} \\
\bar{u}^2_{a,b+\frac{1}{2}} &= \frac{1}{3} u^2_{i-1,j+\frac{3}{2}} + \frac{1}{3} u^2_{i,j+\frac{3}{2}} + \frac{1}{3} u^2_{i+1,j+\frac{3}{2}} \\
\bar{u}^2_{a,b-\frac{1}{2}} &= \frac{1}{3} u^2_{i-1,j-\frac{3}{2}} + \frac{1}{3} u^2_{i,j-\frac{3}{2}} + \frac{1}{3} u^2_{i+1,j-\frac{3}{2}}.
\end{aligned} \tag{C.3}$$

The coarse-grid divergence reads

$$(\bar{D}\bar{u})_{a,b} = (\bar{\delta}_1 \bar{u}^1)_{a,b} + (\bar{\delta}_2 \bar{u}^2)_{a,b}. \tag{C.4}$$

For the first term, we get

$$(\bar{\delta}_1 \bar{u}^1)_{a,b} = \frac{\bar{u}^1_{a+\frac{1}{2},b} - \bar{u}^1_{a-\frac{1}{2},b}}{\bar{\Delta}^1}$$

$$= \frac{1}{3} \frac{u^1_{i+\frac{3}{2},j-1} - u^1_{i-\frac{3}{2},j-1}}{3\Delta^1} + \frac{1}{3} \frac{u^1_{i+\frac{3}{2},j} - u^1_{i-\frac{3}{2},j}}{3\Delta^1} + \frac{1}{3} \frac{u^1_{i+\frac{3}{2},j+1} - u^1_{i-\frac{3}{2},j+1}}{3\Delta^1}$$

$$= \frac{1}{9\Delta^1} \left(u^1_{i+\frac{3}{2},j-1} - (u^1_{i+\frac{1}{2},j-1} - u^1_{i+\frac{1}{2},j-1}) - (u^1_{i-\frac{1}{2},j-1} - u^1_{i-\frac{1}{2},j-1}) - u^1_{i-\frac{3}{2},j-1} \right)$$

$$+ \frac{1}{9\Delta^1} \left(u^1_{i+\frac{3}{2},j} - (u^1_{i+\frac{1}{2},j} - u^1_{i+\frac{1}{2},j}) - (u^1_{i-\frac{1}{2},j} - u^1_{i-\frac{1}{2},j}) - u^1_{i-\frac{3}{2},j} \right) \tag{C.5}$$

$$+ \frac{1}{9\Delta^1} \left(u^1_{i+\frac{3}{2},j+1} - (u^1_{i+\frac{1}{2},j+1} - u^1_{i+\frac{1}{2},j+1}) - (u^1_{i-\frac{1}{2},j+1} - u^1_{i-\frac{1}{2},j+1}) - u^1_{i-\frac{3}{2},j+1} \right)$$

$$= \frac{1}{9} \Big((\delta_1 u^1)_{i+1,j-1} + (\delta_1 u^1)_{i,j-1} + (\delta_1 u^1)_{i-1,j-1}$$

$$+ (\delta_1 u^1)_{i+1,j} + (\delta_1 u^1)_{i,j} + (\delta_1 u^1)_{i-1,j}$$

$$+ (\delta_1 u^1)_{i+1,j+1} + (\delta_1 u^1)_{i,j+1} + (\delta_1 u^1)_{i-1,j+1} \Big).$$

With a similar derivation for $\bar{\delta}_2 \bar{u}^2$, we get the following expression for the coarse grid divergence:

$$(\bar{D}\bar{u})_{a,b} = \frac{1}{9} \Big((Du)_{i-1,j-1} + (Du)_{i,j-1} + (Du)_{i+1,j-1}$$

$$+ (Du)_{i-1,j} + (Du)_{i,j} + (Du)_{i+1,j}$$

$$+ (Du)_{i-1,j+1} + (Du)_{i,j+1} + (Du)_{i+1,j+1} \Big) \tag{C.6}$$

$$= 0.$$

The filtered velocity field is indeed divergence-free.

It is also interesting to point out that $\bar{D}\bar{u}$ can be seen as a volume average of Du, i.e. $\bar{D}\bar{u} = \Psi Du$, or $\bar{D}\Phi = \Psi D$, for a certain pressure filter Ψ built with uniform 3×3-stencils of weights $1/9$. In other words: the face-averaging velocity filter goes hand in hand with a volume-averaging pressure filter.

C.2. Proof for the general case

A similar proof can be shown for a general non-uniform grid in 2D or 3D. Consider a coarse grid index J. The fine grid volumes Ω_I contained inside $\bar{\Omega}_J$ are indexed by $I \in \mathcal{K}_J = \{I \mid \Omega_I \subset \bar{\Omega}_J\}$. The face-averaging filter is defined by

$$\bar{u}^\alpha_J = \sum_{I \in \mathcal{F}^\alpha_J} \rho^\alpha_{J,I} u^\alpha_I, \tag{C.7}$$

where $\mathcal{F}^\alpha_J = \{I \mid \Gamma^\alpha_I \in \bar{\Gamma}^\alpha_J\}$ contains the face-indices and $\rho^\alpha_{J,I}$ are weights to be determined. We assume that $\rho^\alpha_{J,I}$ is independent of I_α and J_α. The fine grid divergence is given by

$$(Du)_I = \sum_{\alpha=1}^d (\delta_\alpha u^\alpha)_I = \sum_{\alpha=1}^d \frac{u^\alpha_{I+h_\alpha} - u^\alpha_{I-h_\alpha}}{|\Delta^\alpha_{I_\alpha}|} = 0. \tag{C.8}$$

The coarse grid divergence is given by

$$(\bar{D}\bar{u})_J = \sum_{\alpha=1}^d (\bar{\delta}_\alpha \bar{u}^\alpha)_J$$

$$= \sum_{\alpha=1}^d \frac{\bar{u}^\alpha_{J+h_\alpha} - \bar{u}^\alpha_{J-h_\alpha}}{|\bar{\Delta}^\alpha_{J_\alpha}|} \quad \text{by definition of finite difference operator}$$

$$= \sum_{\alpha=1}^d \frac{1}{|\bar{\Delta}^\alpha_{J_\alpha}|} \left(\sum_{I \in \mathcal{F}^\alpha_{J+h_\alpha}} \rho^\alpha_{J,I} u^\alpha_I - \sum_{I \in \mathcal{F}^\alpha_{J-h_\alpha}} \rho^\alpha_{J,I} u^\alpha_I \right) \quad \text{by definition of } \bar{u}^\alpha_J$$

$$= \sum_{\alpha=1}^d \frac{1}{|\bar{\Delta}^\alpha_{J_\alpha}|} \sum_{I \in \mathcal{K}_J} \left(\rho^\alpha_{J,I+h_\alpha} u^\alpha_{J,I+h_\alpha} - \rho^\alpha_{J,I-h_\alpha} u^\alpha_{J,I-h_\alpha} \right) \quad \text{telescoping sum over } I_\alpha$$

$$= \sum_{\alpha=1}^d \frac{1}{|\bar{\Delta}^\alpha_{J_\alpha}|} \sum_{I \in \mathcal{K}_J} \rho^\alpha_{J,I} \left(u^\alpha_{I+h_\alpha} - u^\alpha_{I-h_\alpha} \right) \quad \text{since } \rho^\alpha_{J,I} \text{ is independent of } I_\alpha \tag{C.9}$$

$$= \sum_{I \in \mathcal{K}_J} \frac{|\Omega_I|}{|\bar{\Omega}_J|} \sum_{\alpha=1}^{d} \frac{|\bar{\Gamma}_J^{\alpha}|}{|\Gamma_I^{\alpha}|} \rho_{J,I}^{\alpha} \frac{u_{I+h_{\alpha}}^{\alpha} - u_{I-h_{\alpha}}^{\alpha}}{|\Delta_{I_{\alpha}}^{\alpha}|} \quad \text{rewrite terms with } |\Omega_I| = |\Gamma_I^{\alpha}||\Delta_I^{\alpha}| \; \forall \alpha$$

$$= \sum_{I \in \mathcal{K}_J} \frac{|\Omega_I|}{|\bar{\Omega}_J|} (Du)_I \quad \text{if we choose } \rho_{J,I}^{\alpha} = |\Gamma_I^{\alpha}|/|\bar{\Gamma}_J^{\alpha}|$$

$$= 0 \quad \text{since } (Du)_I = 0.$$

The chosen $\rho_{J,I}^{\alpha}$ is indeed independent of I_{α} and J_{α}. We also get the property $\sum_{I \in \mathcal{F}_J^{\alpha}} \rho_{J,I}^{\alpha} = 1$, so constant fine grid velocities are preserved upon filtering. In other words, choosing $\bar{u}_J^{\alpha}$ as a weighted average of the DNS-velocities passing through the coarse volume face $\bar{\Gamma}_J^{\alpha}$ gives a divergence-free $\bar{u}$. Note that the divergence constraint only holds for the face-size weights chosen above. Using arbitrary weights such as Gaussian weights would not work.

We also observe that for the general case, just like for the simple case in Appendix C.1, we can write $\bar{D}\Phi = \Psi D$ for a certain pressure filter Ψ. It is defined by

$$(\Psi p)_J = \sum_{I \in \mathcal{K}_J} \frac{|\Omega_I|}{|\bar{\Omega}_J|} p_I \tag{C.10}$$

for all fields p defined in the pressure points. If we consider the 3×3 compression from Appendix C.1, with $J = (a,b)$, we effectively get $\mathcal{K}_{a,b} = \{i-1,i,i+1\} \times \{j-1,j,j+1\}$ and $|\Omega_I|/|\bar{\Omega}_J| = 1/9$ for all $I \in \mathcal{K}_J$.

Appendix D. Discretize-then-filter (without differentiating the constraint)

In this appendix we show a problem that arises when discretely filtering the differential-algebraic system (3)-(4), instead of filtering the "pressure-free" equation (7). Since the discrete DNS system (3)-(4) includes a divergence term and pressure term, we need to define a pressure-filter Ψ (in addition to the velocity filter Φ), such that $\bar{p} = \Psi p$. This results in the following set of equations:

$$\Psi D u = 0, \tag{D.1}$$

$$\frac{d\bar{u}}{dt} = \Phi F(u) - \Phi G p. \tag{D.2}$$

These can be rewritten as follows:

$$\bar{D}\bar{u} = c_D(u, \bar{u}), \tag{D.3}$$

$$\frac{d\bar{u}}{dt} = \bar{F}(\bar{u}) + \tilde{c}(u, \bar{u}) - \bar{G}\bar{p} + c_P(p, \bar{p}). \tag{D.4}$$

Here $c_D(u, \bar{u}) := (\bar{D}\Phi - \Psi D)u$ represents the commutator error between the discrete divergence operator and filtering, $\tilde{c}(u, \bar{u}) := (\Phi F(u) - \bar{F}(\bar{u}))$ represents the commutator error arising from filtering $F(u)$ (note that it is different from the one used in equation (12)), and the commutator error for the pressure is $c_P(p, \bar{p}) = (\bar{G}\Psi - \Phi G)p$. In case of a face-averaging filter, one has $c_D = 0$, which is the constraint that needs to be enforced by the filtered pressure, when moving to the LES equations. However, in the above formulation an additional commutator error for the pressure appears, which is unwanted and it is unclear how it should be modeled. In the discretize-differentiate-filter approach, this issue with the pressure is circumvented.

Appendix E. Continuous filters and transfer functions

The discrete volume-averaging filter Φ^{VA} is an approximation to the continuous top-hat volume-averaging filter g defined by

$$\begin{aligned}(g * \varphi)(x) &= \int_{\mathbb{R}^d} g(y)\varphi(x-y)\,d\Omega(y) \\ &= \frac{1}{\bar{\Delta}^d} \int_{\left[-\frac{\bar{\Delta}}{2}, \frac{\bar{\Delta}}{2}\right]^d} \varphi(x-y)\,d\Omega(y),\end{aligned} \tag{E.1}$$

where φ is a scalar field,

$$g(x) = \frac{1}{\bar{\Delta}^d} \prod_{\alpha=1}^{d} I\left(x^{\alpha} \in \left[-\frac{\bar{\Delta}}{2}, \frac{\bar{\Delta}}{2}\right]\right)$$

is the convolutional filter kernel, and $I(a \in A)$ is an indicator function equal to 1 if $a \in A$ and 0 otherwise. The continuous filter g can also be interpreted in terms of its transfer function G, defined by

$$G_k = \prod_{\alpha=1}^{d} \frac{\sin(\pi k_{\alpha} \bar{\Delta}/2)}{\pi k_{\alpha} \bar{\Delta}/2} \tag{E.2}$$
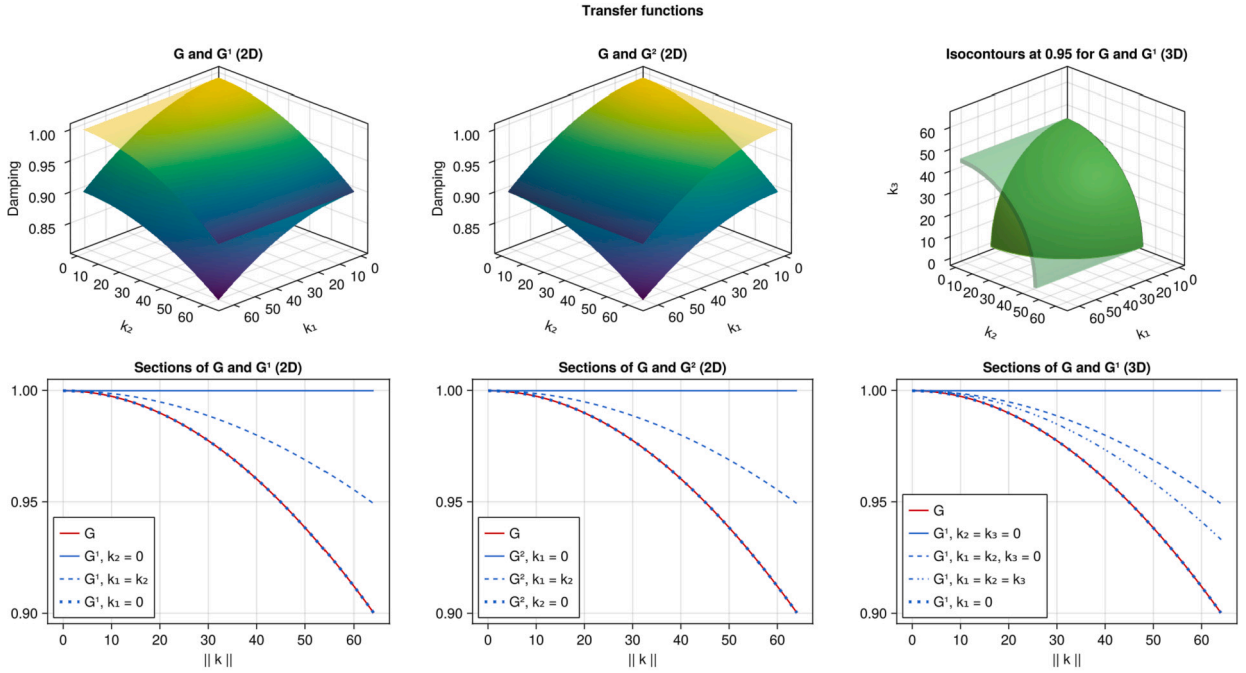
Transfer functions

Fig. E.19. Transfer functions in 2D and 3D. The two top left plots show the 2D transfer functions of the volume-averaging and face-averaging filters. The upper surface is G^α, the lower one is G. The top-right plot shows isocontours of the 3D transfer functions G (behind) and G^1 (transparent, in front) at a given damping level 0.95. The bottom row shows sections of the transfer functions in different wavenumber directions $k/\|k\|$ as a function of the wavenumber magnitude $\|k\|$. G does not depend on the direction.

at a given wavenumber vector $k = (k_1, \ldots, k_d)$. Then the Fourier transform of $g * \varphi$ is given by $\widehat{(g * \varphi)}_k = G_k \hat{\varphi}_k$.

Similarly, we can define the continuous top-hat face-averaging filter (with face normal to the direction α) as

$$(g^\alpha * \varphi)(x) = \int_{\mathbb{R}^d} g^\alpha(y)\varphi(x - y)\,\mathrm{d}\Omega(y), \tag{E.3}$$

where the kernel

$$g^\alpha(x) = \frac{1}{\bar{\Delta}^{d-1}}\delta(x^\alpha)\prod_{\beta \neq \alpha} I\left(x^\beta \in \left[-\frac{\bar{\Delta}}{2}, \frac{\bar{\Delta}}{2}\right]\right) \tag{E.4}$$

is the same as for the volume-averaging filter, but with one interval indicator replaced by a Dirac delta function δ. The transfer function of the face-averaging filter is given by

$$G_k^\alpha = \prod_{\beta \neq \alpha} \frac{\sin(\pi k_\beta \bar{\Delta}/2)}{\pi k_\beta \bar{\Delta}/2}. \tag{E.5}$$

In other words, the α-face-averaging filter is performing top-hat averaging in every direction except for direction α, where it leaves signals intact.

The transfer functions G, G^1, and G^2 are shown in Fig. E.19. The filter width is set to $\bar{\Delta} = 1/128$, corresponding to a cut-off wavenumber level at $\|k\|_\infty = 64$. We also show the transfer functions G and G^1 for the 3D versions of the filters. In 3D, the isocontour of G at a given damping level takes the shape of a sphere, as G_k only depends on $\|k\|$. The isocontour of G^α takes the shape of a cylinder with axis along the α-direction, as the value of G_k^α does not depend on k_α.

We can clearly see that for *all* wavenumbers k, the volume-averaging filter is damping more than the face-averaging filter ($\forall k, G_k \leq G_k^\alpha$). For $k_\alpha = 0$, we get $G_k = G_k^\alpha$ (VA and FA coincide), and for $k_\alpha > 0$, we get the strict inequality $G_k < G_k^\alpha$. For all k such $k_\beta = 0$ for $\beta \neq \alpha$, we get $G_k^\alpha = 1$ (FA does not filter in the normal direction).

Appendix F. Discrete and continuous commutator errors for the 2D Taylor-Green vortex

In this section, we compare the discrete and continuous convective commutator errors. For this purpose we employ an analytical solution to the Navier-Stokes equations on a periodic domain $\Omega = [0, 2\pi]^2$, being the Taylor-Green vortex [71]:

$$u(x, y, t) = -\sin(x)\cos(y)e^{-2vt},$$
$$v(x, y, t) = \cos(x)\sin(y)e^{-2vt}, \tag{F.1}$$
$$p(x, y, t) = \frac{1}{4}(\cos(2x) + \cos(2y))e^{-4vt}.$$

In the remainder of this appendix, u, v, and p will refer to this solution only, and the results that follow are specific to this solution. We will also drop the time dependence and consider $u(x, y) = u(x, y, 0)$ and similarly for v and p.

F.1. Top-hat filter

First, we note that

$$\int\limits_{x-\Delta/2}^{x+\Delta/2} \sin(x')\,dx' = 2\sin(x)\sin(\Delta/2), \qquad \int\limits_{x-\Delta/2}^{x+\Delta/2} \cos(x')\,dx' = 2\cos(x)\sin(\Delta/2). \tag{F.2}$$

Applying a volume-averaging top-hat filter $\varphi \mapsto \bar\varphi$ of width Δ with

$$\bar\varphi(x, y) = \frac{1}{\Delta^2} \int\limits_{x-\Delta/2}^{x+\Delta/2} \int\limits_{y-\Delta/2}^{y+\Delta/2} \varphi(x', y')\,dx'dy' \tag{F.3}$$

gives the following filtered solution:

$$\bar u(x, y) = -\frac{1}{\Delta^2} \int\limits_{x-\Delta/2}^{x+\Delta/2} \sin(x')\,dx' \int\limits_{y-\Delta/2}^{y+\Delta/2} \cos(y')\,dy' = \text{sinc}^2(\Delta/2)u(x, y), \tag{F.4}$$

where $\text{sinc}(x) = \sin(x)/x$ is the transfer function of the 1D top-hat filter. A similar derivation for $\bar v$ and $\bar p$ gives

$$\bar u = \text{sinc}^2(\Delta/2)u, \quad \bar v = \text{sinc}^2(\Delta/2)v, \quad \bar p = \text{sinc}(\Delta)p. \tag{F.5}$$

F.2. Continuous convective commutator error

For the continuous commutator error in the Navier-Stokes equations, we need the nonlinearities $\bar u \bar u$, $\bar u \bar v$, $\bar v \bar u$, and $\bar v \bar v$. They are given by

$$\bar u \bar u = \frac{1}{4}\text{sinc}^4(\Delta/2)(1 - \cos(2x) + \cos(2y) - \cos(2x)\cos(2y)),$$
$$\bar v \bar v = \frac{1}{4}\text{sinc}^4(\Delta/2)(1 + \cos(2x) - \cos(2y) - \cos(2x)\cos(2y)), \tag{F.6}$$
$$\bar u \bar v = -\frac{1}{4}\text{sinc}^4(\Delta/2)\sin(2x)\sin(2y).$$

Next, we are interested in the quantities $\overline{uu}$, $\overline{uv}$, $\overline{vu}$, and $\overline{vv}$. Integrating uu, vv, and uv over x and y separately gives

$$\overline{uu} = \frac{1}{4}(1 - \text{sinc}(\Delta)(\cos(2x) - \cos(2y)) - \text{sinc}^2(\Delta)\cos(2x)\cos(2y)),$$
$$\overline{vv} = \frac{1}{4}(1 + \text{sinc}(\Delta)(\cos(2x) - \cos(2y)) - \text{sinc}^2(\Delta)\cos(2x)\cos(2y)), \tag{F.7}$$
$$\overline{uv} = -\frac{1}{4}\text{sinc}^2(\Delta)\sin(2x)\sin(2y).$$

Finally, we can assemble the sub-filter stress tensor τ:

$$\begin{aligned}
\tau_{xx} &= \overline{uu} - \bar u \bar u \\
&= \frac{1}{4}\Big(1 - \text{sinc}^4(\Delta/2) + (\text{sinc}^4(\Delta/2) - \text{sinc}(\Delta))(\cos(2x) - \cos(2y)) \\
&\quad + (\text{sinc}^4(\Delta/2) - \text{sinc}^2(\Delta))\cos(2x)\cos(2y)\Big), \\
\tau_{xy} &= \overline{uv} - \bar u \bar v \\
&= \frac{1}{4}(\text{sinc}^4(\Delta/2) - \text{sinc}^2(\Delta))\sin(2x)\sin(2y),
\end{aligned} \tag{F.8}$$

and similarly for τ_{yy}. Finally, the sub-filter force terms $c_x = \frac{\partial \tau_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y}$ and $c_y = \frac{\partial \tau_{yx}}{\partial x} + \frac{\partial \tau_{yy}}{\partial y}$ are given by

$$c_x = -\frac{1}{2}(\operatorname{sinc}^4(\Delta/2) - \operatorname{sinc}(\Delta))\sin(2x),$$
$$c_y = -\frac{1}{2}(\operatorname{sinc}^4(\Delta/2) - \operatorname{sinc}(\Delta))\sin(2y).$$

(F.9)

Since the filter width is small compared to the domain size (2π), we can assume that high powers of Δ may be negligible. Using the Taylor series expansions

$$\operatorname{sinc}(x) = 1 - \frac{1}{6}x^2 + \frac{1}{120}x^4 + \mathcal{O}(x^6),$$
$$\operatorname{sinc}^4(x) = 1 - \frac{2}{3}x^2 + \frac{1}{5}x^4 + \mathcal{O}(x^6),$$

(F.10)

we get the coefficient

$$\operatorname{sinc}^4(\Delta/2) - \operatorname{sinc}(\Delta) = \frac{1}{240}\Delta^4 + \mathcal{O}(\Delta^6).$$

(F.11)

If we drop the terms in $\mathcal{O}(\Delta^6)$ and higher, we get the following simplified expressions for the continuous commutator error:

$$c_x = -\frac{\Delta^4}{480}\sin(2x) + \mathcal{O}(\Delta^6),$$
$$c_y = -\frac{\Delta^4}{480}\sin(2y) + \mathcal{O}(\Delta^6).$$

(F.12)

F.3. Discrete convective commutator error

Next, we consider the discrete commutator error computed from a discrete DNS. Assume for simplicity that the DNS field is the exact solution sampled at the DNS grid points. Consider a grid point (x, y). Let d denote the DNS grid spacing.

F.3.1. Discrete filter

Assume that $\Delta/2 = nd$ for some integer n. A simple discretization of the top-hat volume-averaging filter at a coarse grid point (x, y) is given by

$$(\Phi u)(x, y) = \frac{1}{(2n+1)^2}\sum_{i=-n}^{n}\sum_{j=-n}^{n}u(x+id, y+jd)$$
$$= -\frac{1}{(2n+1)^2}\sum_{i=-n}^{n}\sin(x+id)\sum_{j=-n}^{n}\cos(y+jd),$$

(F.13)

and similarly for Φv and Φp. For the x-part, we get

$$\sum_{i=-n}^{n}\sin(x+id) = \sum_{i=-n}^{n}(\sin(x)\cos(id) + \cos(x)\sin(id)) = \left(\sum_{i=-n}^{n}\cos(id)\right)\sin(x).$$

(F.14)

A similar derivation for the y-part, v and p give

$$\Phi u = G_{n,d}^2 u, \quad \Phi v = G_{n,d}^2 v, \quad \Phi p = G_{n,2d}p,$$

(F.15)

where $G_{n,d} = \frac{1}{2n+1}\sum_{i=-n}^{n}\cos(id)$ is the transfer function of the discrete top-hat filter in 1D.

F.3.2. Convective term

We now compute the discrete convective term $\operatorname{conv}_{x,d}(u, v)$ which contributes to the equation for u. This term is required at a u-velocity grid point (x, y), and we thus need the quantity uu at $(x+d/2, y)$ and $(x-d/2, y)$, and the quantity uv at $(x, y+d/2)$ and $(x, y-d/2)$. Knowing that the native points of v are $(x \pm d/2, y \pm d/2)$ when (x, y) is the u-point, we proceed through interpolation.

$A_x(u)$ is obtained by interpolating u left and right:

$$A_x(u)(x+d/2, y) = \frac{1}{2}(u(x, y) + u(x+d, y))$$
$$= -\frac{1}{2}(\sin(x)\cos(y)(1 + \cos(d)) + \cos(x)\cos(y)\sin(d)),$$

(F.16)

$A_x(v)$ is obtained by interpolating v left and right:

$$A_x(v)(x, y + d/2) = \frac{1}{2}(v(x - d/2, y + d/2) + v(x + d/2, y + d/2))$$
$$= \frac{1}{2}\cos(x)(\sin(y)(1 + \cos(d)) + \cos(y)\sin(d)), \tag{F.17}$$

$A_y(u)$ is obtained by interpolating u up and down:

$$A_y(u)(x, y + d/2) = \frac{1}{2}(u(x, y) + u(x, y + d))$$
$$= -\frac{1}{2}\sin(x)(\cos(y)(1 + \cos(d)) - \sin(y)\sin(d)). \tag{F.18}$$

Now we can assemble the quadratic terms.

$$(A_x(u)A_x(u))(x + d/2, y) = \frac{1}{16}((1 - \cos(2x))(1 + \cos(2y))(1 + \cos(d))^2$$
$$+ 2\sin(2x)(1 + \cos(2y))\sin(d)(1 + \cos(d))$$
$$+ (1 + \cos(2x))(1 + \cos(2y))\sin(d)^2), \tag{F.19}$$

$$(A_y(u)A_x(v))(x, y + d/2) = -\frac{1}{16}\sin(2x)\sin(2y)(1 + 2\cos(d) + \cos(2d))$$
$$- \frac{1}{8}\sin(2x)\cos(2y)\sin(d)(1 + \cos(d)). \tag{F.20}$$

The discrete convective term at a point (x, y) is defined as

$$C_{xx,d} = \frac{1}{d}((A(u)A(u))(x + d/2, y) - (A(u)A(u))(x - d/2, y))$$
$$= \frac{1}{4d}\sin(2x)(1 + \cos(2y))\sin(d)(1 + \cos(d)), \tag{F.21}$$

$$C_{xy,d} = \frac{1}{d}\left((A_y(u)A_x(v))(x, y + d/2) - (A_y(u)A_x(v))(x, y - d/2)\right)$$
$$= -\frac{1}{4d}\sin(2x)\cos(2y)\sin(d)(1 + \cos(d)), \tag{F.22}$$

$$\mathrm{conv}_{x,d}(u, v) = C_{xx,d} + C_{xy,d} = \frac{1}{4}\sin(2x)(\mathrm{sinc}(d) + \mathrm{sinc}(2d)). \tag{F.23}$$

A similar derivation for the y-component gives

$$\mathrm{conv}_{y,d}(u, v) = C_{yx,d} + C_{yy,d} = \frac{1}{4}\sin(2y)(\mathrm{sinc}(d) + \mathrm{sinc}(2d)). \tag{F.24}$$

Using the Taylor series expansion (F.10) for the expression

$$\mathrm{sinc}(d) + \mathrm{sinc}(2d) = 2 - \frac{5}{6}d^2 + \frac{17}{120}d^4 + \mathcal{O}(d^6), \tag{F.25}$$

and remembering that the continuous convective x-term is $\frac{1}{2}\sin(2x)$, we get

$$\mathrm{conv}_{x,d}(u, v) = \frac{\partial}{\partial x}(uu) + \frac{\partial}{\partial y}(uv) + \mathcal{O}(d^2), \tag{F.26}$$

and similarly for the y-component. This is just the confirmation that the computation of the derivatives is second order accurate.

Next, we compute the filtered convective force $\Phi\,\mathrm{conv}_x(u, v)$. We get

$$\Phi\,\mathrm{conv}_x(u, v) = \frac{1}{(2n+1)^2}\sum_{i=-n}^{n}\sum_{j=-n}^{n}\frac{1}{4}\sin(2(x + id))(\mathrm{sinc}(d) + \mathrm{sinc}(2d))$$
$$= G_{n,2d}\,\mathrm{conv}_{x,d}(u, v), \tag{F.27}$$

where the transfer function is now applied on $2d$ instead of d. Noting that $\Phi u = G_{n,d}^2 u$ and $\Phi v = G_{n,d}^2 v$ everywhere on the domain, we get

$$A_{x,\Delta}(\Phi u)A_{x,\Delta}(\Phi u) = G_{n,d}^4 A_{x,\Delta}(u)A_{x,\Delta}(u),$$
$$A_{y,\Delta}(\Phi u)A_{x,\Delta}(\Phi v) = G_{n,d}^4 A_{x,\Delta}(u)A_{y,\Delta}(v),$$
$$\mathrm{conv}_{x,\Delta}(\Phi u, \Phi v) = G_{n,d}^4\,\mathrm{conv}_{x,\Delta}(u, v) = \frac{1}{4}G_{n,d}^4(\mathrm{sinc}(\Delta) + \mathrm{sinc}(2\Delta))\sin(2x). \tag{F.28}$$
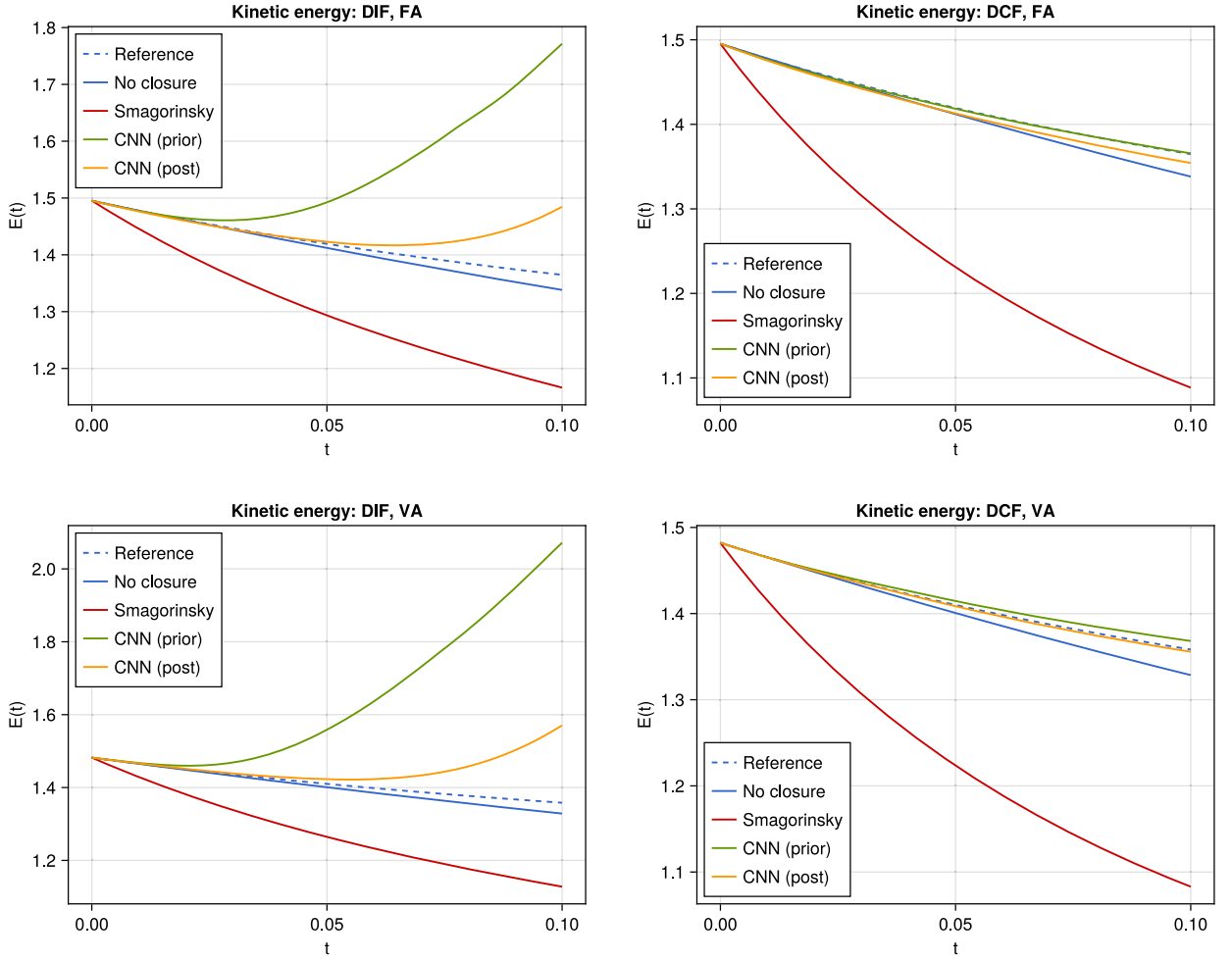
Fig. G.20. Total kinetic energy evolution for the 2D decaying turbulence case at $\bar{n} = 128$. **Left:** Unprojected closure model $\mathcal{M}_{\text{DIF}}$. **Right:** Constrained model $\mathcal{M}_{\text{DCF}}$. **Top:** Face-averaging filter. **Bottom:** Volume-averaging filter.

Using a similar derivation for the y-component, we can now compute the *discrete* convective commutator error

$$
\begin{aligned}
\Phi \operatorname{conv}_{x,d}(u, v) - \operatorname{conv}_{x,\Delta}(\Phi u, \Phi v) &= -E \frac{1}{4} \sin(2x), \\
\Phi \operatorname{conv}_{y,d}(u, v) - \operatorname{conv}_{y,\Delta}(\Phi u, \Phi v) &= -E \frac{1}{4} \sin(2y),
\end{aligned}
\tag{F.29}
$$

where

$$
E = G_{n,d}^4 (\operatorname{sinc}(\Delta) + \operatorname{sinc}(2\Delta)) - G_{n,2d}(\operatorname{sinc}(d) + \operatorname{sinc}(2d))
\tag{F.30}
$$

is the commutator coefficient.

It is important to note that at any given point (x, y), the value of the discrete and continuous convective commutator errors (F.29) and (F.9) are different. The dependence on (x, y) is the same, but the coefficient in front is different. If a closure model is imposed using the traditional route of "filtering first", but trained using commutator error target data computed using DNS ("discretizing first"), there is an inconsistency between the model and the learning environment.

Appendix G. Additional numerical experiments

We here include some additional numerical experiments to confirm that our methods work in a variety of settings.
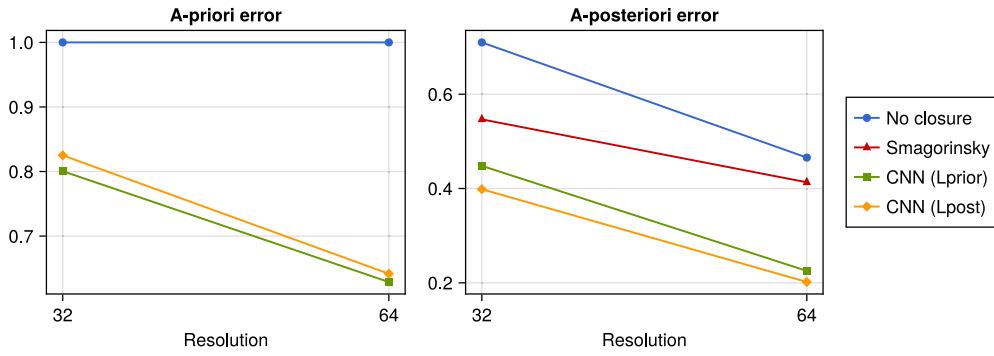
Fig. G.21. A-priori and a-posteriori errors at time $t = 0.29$ for the 3D case.
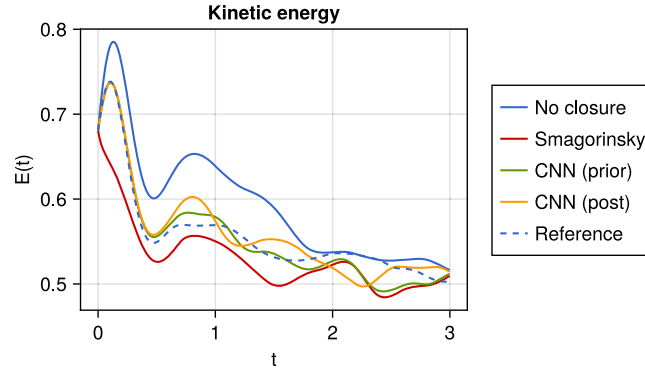

Fig. G.22. Energy evolution for the 3D case with $\bar{n} = 64$.

G.1. LES of decaying turbulence (2D)

We repeat the experiment from 5.2, but without the body force. This results in a decaying turbulence setup, where energy dissipates over time.

Fig. G.20 shows the total kinetic energy evolution for the decaying turbulence test case. Since our discretization is energy-conserving, the DNS energy cannot increase in the absence of a body force, and it can only decrease due to dissipation. If D_2 denotes the discrete diffusion operator, it can be shown that the kinetic energy $E = \frac{1}{2}\|u\|_\Omega^2 = \frac{1}{2}\langle u, u\rangle_\Omega$ satisfies

$$\frac{\mathrm{d}E}{\mathrm{d}t} = \langle u, D_2 u\rangle_\Omega. \tag{G.1}$$

For $\mathcal{M}_{\mathrm{DCF}}$, all models produce decaying energy profiles, but the CNN energy is the most accurate. The no-closure model is slightly too dissipative, and the Smagorinsky model is even more dissipative. For $\mathcal{M}_{\mathrm{DIF}}$, the no-closure and Smagorinsky models have similar profiles as for $\mathcal{M}_{\mathrm{DCF}}$, but the CNN models become unstable. Training with the a-posteriori loss function corrects for this instability, leading to correct energy levels for a longer time, but the instability is still present.

G.2. LES of forced turbulence (3D)

To show that the formulation also works in 3D, we consider a decaying turbulence test case in a periodic box $\Omega = [0, 1]^3$ with $N = 1024^3$ finite volumes. We use single precision floating point numbers, Reynolds number $\mathrm{Re} = 2000$, simulation time $t_{\mathrm{end}} = 3$, burn-in time $t_{\mathrm{burn}} = 0.5$, and 10 random initial conditions. Unlike for the extensive analysis in section 5.2, we only consider 2 LES resolutions $\bar{n} \in \{32, 64\}$, one filter (face-averaging), and one LES model ($\mathcal{M}_{\mathrm{DCF}}$). We use the same body force as in the 2D case, namely f with $f^\alpha(x) = 5\delta_{\alpha=1}\sin(8\pi x^2)$. We use the same CNN architecture as in the 2D case, but with a kernel size $5 \times 5 \times 5$ instead of 5×5, resulting in 234 096 learnable parameters instead of 45 696. For the Smagorinsky model, we use the common choice of $\theta = 0.17$.

Fig. G.21 shows the a-priori and a-posteriori errors for the 3D case at time $t = 0.29$. As in the 2D case (Figs. 10 and 11), the a-priori trained CNN produces lower a-priori errors, while the a-posteriori trained CNN produces lower a-posteriori errors. For both training methods, the CNN performs better than the no-closure model and Smagorinsky.

Fig. G.22 shows the energy evolution in the 3D case. As in the 2D case (Fig. 12), the CNN energy stays close to the target energy much longer than the no-closure and Smagorinsky energies. Additionally, the Smagorinsky energy seems to be more dissipative, while for the 2D case this was not visible due to the energy injection from the body force.
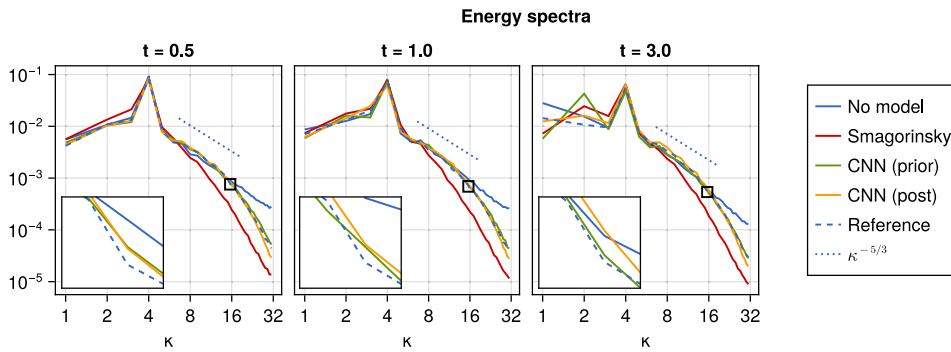
Energy spectra

Fig. G.23. Energy spectra for the 3D case with $\bar{n} = 64$.

Fig. G.23 shows the energy spectra in the 3D case. A clear peak is visible at the body force injection wavenumber at $\kappa = 4$. The Smagorinsky model is clearly too dissipative in the high wavenumbers. The CNN is outperforming both of the two other models, and the spectrum stays close long after the LES solution has become decorrelated from the reference solution, confirming that while the closure model is not able to predict the exact chaotic solution for long times, it is able to track the turbulent statistics.

Data availability

The manuscript contains a link to the software used to generate all results, including random number seeds for reproducibility. It is on GitHub under the MIT License.

References

[1] S.D. Agdestein, B. Sanderse, Discretize first, filter next – a new closure model approach, in: 8th European Congress on Computational Methods in Applied Sciences and Engineering, CIMNE, 2022.

[2] H. Jane Bae, Adrian Lozano-Duran, Numerical and modeling error assessment of large-eddy simulation using direct-numerical-simulation-aided large-eddy simulation, arXiv:2208.02354, 2022.

[3] A.R. Barron, Universal approximation bounds for superpositions of a sigmoidal function, IEEE Trans. Inf. Theory 39 (3) (May 1993) 930–945.

[4] Andrea Beck, David Flad, Claus-Dieter Munz, Deep neural networks for data-driven les closure models, J. Comput. Phys. 398 (2019) 108910.

[5] Andrea Beck, Marius Kurz, A perspective on machine learning methods in turbulence modeling, GAMM-Mitt. 44 (1) (2021) e202100002.

[6] Andrea Beck, Marius Kurz, Toward discretization-consistent closure schemes for large eddy simulation using reinforcement learning, Phys. Fluids 35 (12) (2023) 125122.

[7] Mark Benjamin, Gianluca Iaccarino, Neural network–based closure models for large–eddy simulations with explicit filtering, Available at SSRN, https://ssrn.com/abstract=4462714, 2023.

[8] Luigi C. Berselli, Traian Iliescu, William J. Layton, Mathematics of Large Eddy Simulation of Turbulent Flows, Springer-Verlag, 2006.

[9] Tim Besard, Valentin Churavy, Alan Edelman, Bjorn De Sutter, Rapid software prototyping for heterogeneous and distributed platforms, Adv. Eng. Softw. 132 (2019) 29–46.

[10] Tim Besard, Christophe Foket, Bjorn De Sutter, Effective extensible programming: unleashing Julia on GPUs, IEEE Trans. Parallel Distrib. Syst. (2018).

[11] Jeff Bezanson, Alan Edelman, Stefan Karpinski, Viral B. Shah, Julia: a fresh approach to numerical computing, SIAM Rev. 59 (1) (Sep 2017) 65–98.

[12] A. Báez Vidal, O. Lehmkuhl, F.X. Trias, C.D. Pérez-Segarra, On the properties of discrete spatial filters for CFD, J. Comput. Phys. 326 (2016) 474–498.

[13] Gary J. Chandler, Rich R. Kerswell, Invariant recurrent solutions embedded in a turbulent two-dimensional Kolmogorov flow, J. Fluid Mech. 722 (2013) 554–595.

[14] Tianping Chen, Hong Chen, Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems, IEEE Trans. Neural Netw. 6 (4) (1995) 911–917.

[15] Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, Evan Shelhamer, cudnn: efficient primitives for deep learning, arXiv:1410.0759, 2014.

[16] Alexandre Joel Chorin, Numerical solution of the Navier-Stokes equations, Math. Comput. 22 (104) (1968) 745–762.

[17] Valentin Churavy, Dilum Aluthge, Anton Smirnov, Julian Samaroo, James Schloss, Lucas C. Wilcox, Simon Byrne, Tim Besard, Maciej Waruszewski, Ali Ramadhan, Meredith, Simeon Schaub, Navid C. Constantinou, Jake Bolewski, Max Ng, William Moses, Carsten Bauer, Ben Arthur, Charles Kawczynski, Chris Hill, Christopher Rackauckas, Gunnar Farnebäck, Hendrik Ranocha, James Cook, Jinguo Liu, Michel Schanen, Oliver Schulz, JuliaGPU/KernelAbstractions.jl: v0.9.10, Zenodo, Oct. 2023.

[18] Simon Danisch, Julius Krumbiegel, Makie.jl: flexible high-performance data visualization for Julia, J. Open Sour. Softw. 6 (65) (2021) 3349.

[19] Karthik Duraisamy, Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence, Phys. Rev. Fluids 6 (5) (May 2021).

[20] Karthik Duraisamy, Gianluca Iaccarino, Heng Xiao, Turbulence modeling in the age of data, Annu. Rev. Fluid Mech. 51 (1) (January 2019) 357–377.

[21] Timothy Gallagher, Ayaboe Edoh, Venke Sankaran, Residual and solution filtering for explicitly-filtered large-eddy simulations, in: AIAA Scitech 2019 Forum, American Institute of Aeronautics and Astronautics, Jan. 2019.

[22] Timothy P. Gallagher, Vaidyanathan Sankaran, Affordable explicitly filtered large-eddy simulation for reacting flows, AIAA J. 57 (2) (Feb 2019) 809–823.

[23] Thomas B. Gatski, M. Yousuff Hussaini, John L. Lumley, Simulation and Modeling of Turbulent Flows, Oxford University Press, 1996.

[24] M. Germano, Differential filters for the large eddy numerical simulation of turbulent flows, Phys. Fluids 29 (6) (1986) 1755–1757.

[25] Bernard J. Geurts, Darryl D. Holm, Commutator errors in large-eddy simulation, J. Phys. A, Math. Gen. 39 (9) (February 2006) 2213–2229.

[26] Yifei Guan, Adam Subel, Ashesh Chattopadhyay, Pedram Hassanzadeh, Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate LES, Phys. D: Nonlinear Phenom. 443 (2023).

[27] E. Hairer, G. Wanner, O. Solving, Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems, Springer, 1991.

[28] Ernst Hairer, Christian Lubich, Michel Roche, The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods, vol. 1409, Springer, 2006.

[29] Francis H. Harlow, J. Eddie Welch, Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface, Phys. Fluids 8 (12) (1965) 2182–2189.

[30] Philipp Holl, Vladlen Koltun, Kiwon Um, Nils Thuerey, Phiflow: a differentiable PDE solving framework for deep learning via physical simulations, in: NeurIPS Workshop, vol. 2, 2020.

[31] Michael Innes, Don't unroll adjoint: differentiating SSA-form programs, CoRR, arXiv:1810.07951, 2018.

[32] Jinhee Jeong, Fazle Hussain, On the identification of a vortex, J. Fluid Mech. 285 (1995) 69–94.

[33] Ryan N. King, Peter E. Hamlington, Werner JA Dahm, Autonomic closure for turbulence simulations, Phys. Rev. E 93 (3) (2016) 031301.

[34] Diederik P. Kingma, Jimmy Ba, Adam: a method for stochastic optimization, arXiv:1412.6980v9, 2017.

[35] Dmitrii Kochkov, Jamie A. Smith, Ayya Alieva, Qing Wang, Michael P. Brenner, Stephan Hoyer, Machine learning–accelerated computational fluid dynamics, Proc. Natl. Acad. Sci. 118 (21) (2021) e2101784118.

[36] Georg Kohl, Li-Wei Chen, Nils Thuerey, Turbulent flow simulation using autoregressive conditional diffusion models, arXiv:2309.01745, Sep 2023.

[37] M. Kurz, A. Beck, Investigating model-data inconsistency in data-informed turbulence closure terms, in: 14th WCCM-ECCOMAS Congress, CIMNE, 2021.

[38] Marius Kurz, Andrea Beck, A machine learning framework for LES closure terms, Electron. Trans. Numer. Anal. 56 (2022) 117–137, arXiv:2010.03030v1, Oct. 2020.

[39] D.K. Lilly, On the numerical simulation of buoyant convection, Tellus 14 (2) (1962) 148–172.

[40] D.K. Lilly, The representation of small-scale turbulence in numerical simulation experiments, in: Proc. IBM Sci. Comput. Symp. on Environmental Science, 1967, pp. 195–210.

[41] Douglas K. Lilly, On the computational stability of numerical solutions of time-dependent non-linear geophysical fluid dynamics problems, Mon. Weather Rev. 93 (1) (1965) 11–25.

[42] Bjoern List, Li-Wei Chen, Kartik Bali, Nils Thuerey, Differentiability in unrolled training of neural physics simulators on transient dynamics, Comput. Methods Appl. Mech. Eng. 433 (2025) 117441.

[43] Björn List, Li-Wei Chen, Nils Thuerey, Learned turbulence modelling with differentiable fluid solvers: physics-based loss functions and optimisation horizons, J. Fluid Mech. 949 (2022) A25.

[44] Ilya Loshchilov, Frank Hutter, Sgdr: stochastic gradient descent with warm restarts, arXiv:1608.03983, 2017.

[45] David G. Luenberger, The conjugate residual method for constrained minimization problems, SIAM J. Numer. Anal. 7 (3) (Sep 1970) 390–398.

[46] T.S. Lund, The use of explicit filters in large eddy simulation, in: Turbulence Modelling and Simulation, Comput. Math. Appl. 46 (4) (2003) 603–616.

[47] Jonathan F. MacArt, Justin Sirignano, Jonathan B. Freund, Embedded training of neural-network subgrid-scale turbulence models, Phys. Rev. Fluids 6 (May 2021) 050502.

[48] Romit Maulik, Omer San, A neural network approach for the blind deconvolution of turbulent flows, J. Fluid Mech. 831 (2017) 151–181.

[49] Romit Maulik, Omer San, A stable and scale-aware dynamic modeling framework for subgrid-scale parameterizations of two-dimensional turbulence, in: Large-Eddy and Direct Numerical Simulations of Oceanic and Atmospheric Flows, Comput. Fluids 158 (2017) 11–38.

[50] Romit Maulik, Omer San, Adil Rasheed, Prakash Vedula, Sub-grid modelling for two-dimensional turbulence using neural networks, arXiv:1808.02983v1, Aug 2018.

[51] Hugo Melchers, Daan Crommelin, Barry Koren, Vlado Menkovski, Benjamin Sanderse, Comparison of neural closure models for discretised PDEs, Comput. Math. Appl. 143 (2023) 94–107.

[52] Paolo Orlandi, Fluid Flow Phenomena: a Numerical Toolkit, vol. 55, Springer Science & Business Media, 2000.

[53] Steven A. Orszag, G.S. Patterson, Numerical simulation of three-dimensional homogeneous isotropic turbulence, Phys. Rev. Lett. 28 (Jan 1972) 76–79.

[54] Avik Pal, Lux: Explicit Parameterization of Deep Neural Networks in Julia, April 2023.

[55] Jonghwan Park, Haecheon Choi, Toward neural-network-based large eddy simulation: application to turbulent channel flow, J. Fluid Mech. 914 (March 2021).

[56] J. Blair Perot, Discrete conservation properties of unstructured mesh schemes, Annu. Rev. Fluid Mech. 43 (2011) 299–318.

[57] Stephen B. Pope, Turbulent Flows, Cambridge University Press, Cambridge, England, Aug 2000.

[58] R.S. Rogallo, Numerical experiments in homogeneous turbulence, NASA-TM-81315, 1981.

[59] R.S. Rogallo, P. Moin, Numerical simulation of turbulent flows, Annu. Rev. Fluid Mech. 16 (1) (1984) 99–137.

[60] Pierre Sagaut, Large Eddy Simulation for Incompressible Flows: an Introduction, Springer Science & Business Media, 2005.

[61] Omer San, Anne E. Staples, High-order methods for decaying two-dimensional homogeneous isotropic turbulence, Comput. Fluids 63 (Jun 2012) 105–127.

[62] B. Sanderse, Energy-conserving Runge–Kutta methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 233 (2013) 100–131.

[63] Benjamin Sanderse, Panos Stinis, Romit Maulik, Shady E. Ahmed, Scientific Machine Learning for Closure Models in Multiscale Problems: A Review, March 2024.

[64] Varun Shankar, Vedant Puri, Ramesh Balakrishnan, Romit Maulik, Venkatasubramanian Viswanathan, Differentiable physics-enabled closure modeling for Burgers' turbulence, arXiv:2209.11614v1.

[65] Shyy Wei, A study of finite difference approximations to steady-state, convection-dominated flow problems, J. Comput. Phys. 57 (3) (February 1985) 415–438.

[66] Justin Sirignano, Jonathan MacArt, Konstantinos Spiliopoulos, PDE-constrained models with neural network terms: optimization and global convergence, arXiv: 2105.08633v6, May 2021.

[67] Justin Sirignano, Jonathan F. MacArt, Dynamic deep learning LES closures: online optimization with embedded DNS, arXiv:2303.02338v1, Mar 2023.

[68] Justin Sirignano, Jonathan F. MacArt, Jonathan B. Freund, DPM: a deep learning PDE augmentation method with application to large-eddy simulation, J. Comput. Phys. 423 (2020) 109811.

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

[70] Robin Stoffer, Caspar M. van Leeuwen, Damian Podareanu, Valeriu Codreanu, Menno A. Veerman, Martin Janssens, Oscar K. Hartogensis, Chiel C. van Heerwaarden, Development of a large-eddy simulation subgrid model based on artificial neural networks: a case study of turbulent channel flow, Geosci. Model Dev. 14 (6) (June 2021) 3769–3788.

[71] Geoffrey Ingram Taylor, Albert Edward Green, Mechanism of the production of small eddies from large ones, Proc. R. Soc. Lond. Ser. A, Math. Phys. Sci. 158 (895) (February 1937) 499–521.

[72] F.X. Trias, R.W.C.P. Verstappen, On the construction of discrete filters for symmetry-preserving regularization models, Comput. Fluids 40 (1) (2011) 139–148.

[73] Kiwon Um, Robert Brand, Yun Fei (Raymond), Philipp Holl, Nils Thuerey, Solver-in-the-loop: learning from differentiable physics to interact with iterative PDE-solvers, in: H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, vol. 33, Curran Associates, Inc., 2020, pp. 6111–6122.

[74] Toby van Gastelen, Wouter Edeling, Benjamin Sanderse, Energy-Conserving Neural Network for Turbulence Closure Modeling, February 2023.

[75] R.W.C.P. Verstappen, A.E.P. Veldman, Symmetry-preserving discretization of turbulent flow, J. Comput. Phys. 187 (1) (May 2003) 343–368.

[76] R.W.C.P. Verstappen, A.E.P. Veldman, Direct numerical simulation of turbulence at lower costs, J. Eng. Math. 32 (2/3) (1997) 143–159.

[77] Alan A. Wray, Minimal storage time advancement schemes for spectral methods, Report No. MS, 202, NASA Ames Research Center, California, 1990.