SCIENTIFIC MACHINE LEARNING FOR CLOSURE MODELS IN MULTISCALE PROBLEMS: A REVIEW

B. SANDERSE, P. STINIS, R. MAULIK, AND S. E. AHMED

ABSTRACT. Closure problems are omnipresent when simulating multiscale systems, where some quantities and processes cannot be fully prescribed despite their effects on the simulation's accuracy. Recently, scientific machine learning approaches have been proposed as a way to tackle the closure problem, combining traditional (physics-based) modeling with data-driven (machine-learned) techniques, typically through enriching differential equations with neural networks. This paper reviews the different reduced model forms, distinguished by the degree to which they include known physics, and the different objectives of a priori and a posteriori learning. The importance of adhering to physical laws (such as symmetries and conservation laws) in choosing the reduced model form and choosing the learning method is discussed. The effect of spatial and temporal discretization and recent trends toward discretization-invariant models are reviewed. In addition, we make the connections between closure problems and several other research disciplines: inverse problems, Mori-Zwanzig theory, and multi-fidelity methods. In conclusion, much progress has been made with scientific machine learning approaches for solving closure problems, but many challenges remain. In particular, the generalizability and interpretability of learned models is a major issue that needs to be addressed further.

Keywords: closure model, scientific machine learning, multiscale problem, turbulence

1. INTRODUCTION AND BACKGROUND

Closure problems are everywhere. They arise when a mathematical-physical model that describes certain quantities of interest is created, *which depends on quantities that are not of prime interest but whose effect cannot be neglected.* A prominent example is in numerical weather prediction, where the aim is to predict tomorrow's weather given the physical laws that govern fluid flows and today's conditions. In principle, these physical laws describe all relevant physics, but in practice certain processes like cloud formation take place on such small spatial and temporal scales that resolving (simulating) them on a computer is impossible, even though *their effect on the weather is crucial* [1].

The weather prediction problem is an example of a multiscale problem in which the quantities of interest are associated with large scales, but depend on quantities associated with small scales. If these small scales are not resolved, the large-scale equations are *unclosed.* In this article, we focus on multiscale problems that lead to a closure problem, and in particular we consider multiscale problems that involve a continuum of scales (fluid flows being an important example). There is no clear separation of scales, and the closure model is needed everywhere in the domain to supplement the large-scale model (Type "B" problems in the terminology of E [2]).

Closure models appear under different names in different domains [3]. The term "closure" is most common in the fluid dynamics community, designating the turbulence closure problem—closure models are known as subgrid or subgrid-scale (SGS) models. In kinetic equations and radiation transport, the term "moment closure" is used [4, 5, 6]. In numerical weather prediction and climate science, the problem is known as "parameterization", and finding accurate and stable parameterizations is considered a key challenge [1, 7]. In general circulation models (including atmospheric and oceanic models), the same terminology is used [8, 9]. In the reservoir simulation community, the closure problem is commonly referred to as "homogenization" or "upscaling" [10]. In materials science, molecular dynamics, and computational biology, the term "coarse graining" is used [11, 12].

Finding a closure model for multiscale problems is difficult and often more an art than a science. As mentioned, part of the model description is often known from physical laws (such as conservation laws), and only the effect of the small scales on the large scales needs to be approximated. The mapping associated with this approximation is typically strongly nonlinear—therefore, machine

learning methods form a promising avenue, given their strong approximation properties. Consequently, a hybrid approach to the closure problem has become a popular research field, involving a combination of a physics-based model describing the large scales and a machine-learning based model to approximate the effect of the small scales. Such approaches fall within the realm of what is now known as scientific machine learning. Because the physics-based model typically consists of differential equations, this hybrid approach in practice means a mixture of differential equations with neural networks. This leads to many new and interesting research questions regarding stability, consistency, and convergence, which have only started to be addressed recently.

Consequently, the aim of this review is to give an overview of recent scientific machine learning approaches to the closure problem and the open challenges faced by the community. To do so, we give a somewhat abstract view on the closure problem, which is useful because it provides a generic description that fits several application domains. A second aim of this review is to make the connection to adjacent research disciplines clear; for example, the inverse problem community, the Mori-Zwanzig community, etc. We realize that our notation and the cited literature are somewhat geared toward fluid flow applications (and large eddy simulations [LESs] in particular), but we have tried to make it general enough to address a broad audience. We also note that many excellent review articles have appeared on the topic of machine learning, partial differential equations and/or fluid mechanics, for example [13, 3, 14, 15, 16]. In this review, we therefore assume that the reader is familiar with basic concepts of machine learning and differential equations.

The outline of this paper is as follows. Section 2 and Section 3 describe the fundamentals of closure modeling, splitting the closure problem into two parts: the reduced model form (*what* to learn), and the choice for the objective function (*how* to learn it). Together with Section 4 (embedding physics constraints) and Section 5 (spatial and temporal discretization aspects), these sections describe the methodological framework in which many existing closure model approaches fit. Sections 6, 7 and 8 each give a different perspective on the closure problem from the viewpoint of inverse problems, from the viewpoint of Mori-Zwanzig theory, and from the viewpoint of multi-fidelity and multiscale methods. Section 9 concludes with directions for future research.

2. Reduced model forms

2.1. **Model reduction.** We will consider partial differential equations (PDEs) of the form

$$\boldsymbol{F}(\boldsymbol{u}; \boldsymbol{\mu}) = 0, \tag{1}$$

where $\boldsymbol{F}$ is the PDE operator (containing temporal and spatial derivatives), $\boldsymbol{u}(\boldsymbol{x}, t)$ is the solution, and $\boldsymbol{\mu}$ represents physical parameters or parameterized initial or boundary conditions. Equation (1) will be referred to as the *full model*. In practice, this full model is solved by discretizing in space and time. The resulting discretized model will be referred to as the *high-fidelity model*, assuming a sufficient resolution is maintained to capture the relevant spatial and temporal scales..

The degrees of freedom or spectral content of the solution $\boldsymbol{u}(\boldsymbol{x}, t)$ can be *reduced*, e.g., through filtering, averaging, projection, or truncation. The reduction operator $\mathcal{A}$ leads to a reduced field $\bar{\boldsymbol{u}}(\boldsymbol{x}, t)$:

$$\bar{\boldsymbol{u}} := \mathcal{A}(\boldsymbol{u}). \tag{2}$$

$\mathcal{A}$ is often a linear operator, but this is not necessary. We assume in this review that *the main interest is the computation of an accurate approximation to* $\bar{\boldsymbol{u}}$, not in computing an accurate approximation to $\boldsymbol{u}$. We also assume that a small number of simulations of (1) can be performed, leading to reference data for both the solution field and the reduced field. Note that a further reduction is sometimes made, i.e., in certain applications where one is only interested in a quantity of interest $Q(\bar{\boldsymbol{u}})$ that is of lower dimension than $\bar{\boldsymbol{u}}$ itself. Two straightforward but important relations in the context of model reduction are $\bar{\boldsymbol{F}}(\boldsymbol{u}; \boldsymbol{\mu}) := \mathcal{A}(\boldsymbol{F}(\boldsymbol{u}; \boldsymbol{\mu})) = 0$ [1] and $\boldsymbol{F}(\bar{\boldsymbol{u}}; \boldsymbol{\mu}) \neq 0$. In other words, the reduction operator $\mathcal{A}$ and the PDE operator $\boldsymbol{F}$ do not necessarily commute.

The reduction operator $\mathcal{A}$ can be local in space and time, but also nonlocal (e.g., to account for memory effects). A prominent example of a reduction operator is a filter with a kernel K [17]:

$$\bar{\boldsymbol{u}}(\boldsymbol{x}, t) = \int \boldsymbol{u}(\boldsymbol{\xi}, t) K(\boldsymbol{x}, \boldsymbol{\xi}) \mathrm{d}\boldsymbol{\xi}, \tag{3}$$

---

[1]assuming $\mathcal{A}$ is linear

where the kernel is often of the convolution type, so $K(\boldsymbol{x}, \boldsymbol{\xi}) = K(\boldsymbol{x} - \boldsymbol{\xi})$. Another example of a reduction operator is an expansion in an orthogonal basis $\boldsymbol{\phi}_i(\boldsymbol{x})$ truncated to a finite dimension:

$$(4) \qquad \bar{\boldsymbol{u}}(\boldsymbol{x}, t) = \sum_{i=1}^{N} a_i(t) \boldsymbol{\phi}_i(\boldsymbol{x}),$$

which is commonly employed in the Reduced-Order Model (ROM) community [18, 3].

The main problem is that $\bar{\boldsymbol{u}}$ is in general not a solution to PDE (1). The common approach to predict $\bar{\boldsymbol{u}}$ is to formulate a parameterized *reduced model* $\boldsymbol{G}$ (typically a PDE, but this is not necessary) with parameters $\boldsymbol{\theta}$, which describes an approximation $\bar{\boldsymbol{v}} \approx \bar{\boldsymbol{u}}$ *that does not explicitly depend on* $\boldsymbol{u}$:

$$(5) \qquad \boxed{\boldsymbol{G}_{\boldsymbol{\theta}}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) = 0}.$$

This model $\boldsymbol{G}$ should be computationally much faster to solve than the original PDE (1), while retaining most of the accuracy of the quantity of interest (QoI). We switched notation from $\bar{\boldsymbol{u}}(\boldsymbol{x}, t)$ to $\bar{\boldsymbol{v}}(\boldsymbol{x}, t)$ because solving (5) will in general not yield the reduced field $\bar{\boldsymbol{u}}$, since $\bar{\boldsymbol{u}}$ requires knowledge of the full solution $\boldsymbol{u}$, as defined by (2). In addition, $\bar{\boldsymbol{v}}(\boldsymbol{x}, t)$ could possibly include additional degrees of freedom, such as latent variables, that aid in approximating the evolution of $\bar{\boldsymbol{u}}$. The vector $\boldsymbol{\theta}$ contains parameters or parameterized functions (e.g., neural networks, polynomials, etc.) that need to be learned or discovered, depending on the hypotheses and Ansatz presumed in the problem formulation.

The main challenge lies in deriving an expression for $\boldsymbol{G}$ (including the specification of all necessary parameters), thus placing us in the setting of *model discovery* [19, 20, 21, 22, 23] (even though the term model discovery is often used in the context of discovering the PDE for $\boldsymbol{u}$ instead of for $\bar{\boldsymbol{u}}$). This means that the governing equations are not fully known, and techniques such as physics-informed neural networks (PINNs) [24] and neural finite element methods [25] cannot be easily applied because these assume that the PDE is known (it enters in the loss function). For a review of the development of these methods, see for example [15, 26, 27, 28]. However, because PINNs can also be used to solve inverse problems, they might be effective in detecting closure models if the closure model is framed as an inverse problem—see sections 4.1 and 6.1.

The difficulty of deriving an expression for $\boldsymbol{G}$ is not only due to its inverse-problem nature, but also stems from the fact that the dynamics described by the full model $\boldsymbol{F}$ is often nonlinear and *chaotic*. This means that the full model $\boldsymbol{F}$ would typically be solved with a statistical approach, and a similar strategy can be expected for the reduced model—see Sections 6.4 and 7. Nevertheless, many reduced models used to date are still deterministic.

2.2. Closure problem.
Equation (5) describes a very generic class of models for $\bar{\boldsymbol{v}}$. A natural approach is to find $\boldsymbol{G}$ such that it approximates the reduced model $\bar{\boldsymbol{F}}$, i.e., $\boldsymbol{G} \approx \bar{\boldsymbol{F}}$. Applying the reduction operation to (1) yields:

$$(6) \qquad \bar{\boldsymbol{F}}(\boldsymbol{u}; \boldsymbol{\mu}) := \mathcal{A}(\boldsymbol{F}(\boldsymbol{u}; \boldsymbol{\mu})) = 0 \quad \rightarrow \quad \boldsymbol{F}(\bar{\boldsymbol{u}}; \boldsymbol{\mu}) + \boldsymbol{C}(\boldsymbol{u}, \bar{\boldsymbol{u}}; \boldsymbol{\mu}) = 0$$

where

$$(7) \qquad \boldsymbol{C}(\boldsymbol{u}, \bar{\boldsymbol{u}}; \boldsymbol{\mu}) = \bar{\boldsymbol{F}}(\boldsymbol{u}; \boldsymbol{\mu}) - \boldsymbol{F}(\bar{\boldsymbol{u}}; \boldsymbol{\mu}).$$

This equation shows that one cannot simply solve the original PDE for the reduced field, because the reduction step and the PDE operator do not commute. Generally, the equation for $\bar{\boldsymbol{F}}$ involves $\boldsymbol{u}$ and is therefore *unclosed*. The common expression for $\boldsymbol{G}$ follows from approximating the commutator error $\boldsymbol{C}$ by a model $\boldsymbol{m}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}}; \boldsymbol{\mu})$ that depends only on $\bar{\boldsymbol{u}}$:

$$(8) \qquad \boldsymbol{m}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}}; \boldsymbol{\mu}) \approx \boldsymbol{C}(\boldsymbol{u}, \bar{\boldsymbol{u}}; \boldsymbol{\mu}),$$

so that

$$(9) \qquad \boxed{\boldsymbol{G}_{\boldsymbol{\theta}}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) = \boldsymbol{F}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) + \boldsymbol{m}_{\boldsymbol{\theta}}(\bar{\boldsymbol{v}}; \boldsymbol{\mu})}.$$

The model $\boldsymbol{m}_{\boldsymbol{\theta}}$ is known as the *closure model*. Expression (9) is the basic equation for the reduced model, which expresses that the closure model approximates a commutator error that acts as an

additive correction to the full model. If additional knowledge of $\boldsymbol{F}$ is available, e.g., when $\boldsymbol{F}$ is of the form

$$(10) \qquad \boldsymbol{F}(\boldsymbol{u}; \boldsymbol{\mu}) = \frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{f}(\boldsymbol{u}; \boldsymbol{\mu}),$$

the expression for $\boldsymbol{G}$ reads [2]:

$$(11) \qquad \boxed{\boldsymbol{G_\theta}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) = \frac{\partial \bar{\boldsymbol{v}}}{\partial t} + \boldsymbol{f}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) + \boldsymbol{m_\theta}(\bar{\boldsymbol{v}}; \boldsymbol{\mu})}.$$

In practice, this form is as important as equation (9) because in many physical problems, $\boldsymbol{F}$ is of the form (10). In addition, in fluid flow and material science applications, $\boldsymbol{f}$ usually involves the divergence of a stress tensor, and (11) is further specialized to $\boldsymbol{G_\theta}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) = \frac{\partial \bar{\boldsymbol{v}}}{\partial t} + \boldsymbol{f}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) + \nabla \cdot \hat{\boldsymbol{m}}_{\boldsymbol{\theta}}(\bar{\boldsymbol{v}}; \boldsymbol{\mu})$. An example is given in Section 2.4.

In many scientific machine learning approaches, the term $\boldsymbol{m}$ is modeled by a neural network. This is a logical choice, given that the mapping from $\bar{\boldsymbol{u}}$ to $\boldsymbol{m}(\bar{\boldsymbol{u}})$ is generally highly nonlinear. The resulting PDE consists of "known physics", $\boldsymbol{f}(\bar{\boldsymbol{v}})$, plus "learned physics" (a neural network). However, the stability of this PDE solution is not guaranteed and is a subject of significant research, which will be further discussed in Section 3. In addition, approaches like reinforcement learning provide an alternative perspective on learning $\boldsymbol{m}$ – see section 3.5.

2.3. **Other reduced model forms.** If one removes the condition that $\boldsymbol{G}$ should approximate $\boldsymbol{F}$ and the focus is only on having $\bar{\boldsymbol{v}} \approx \bar{\boldsymbol{u}}$, many other forms for $\boldsymbol{G}$ are possible, typically depending on how much a priori knowledge of $\boldsymbol{F}$ is given. In addition, the degree of code access (e.g., whether the source code is available, or only a compiled version) can determine the form chosen for $\boldsymbol{G}$. These non-closure forms are not the main focus of this paper but will be mentioned for completeness. For example, if the exact form of $\boldsymbol{F}$ is not known or available and one only knows that the reduced model should be of evolution form, one can learn the *entire* right-hand side (instead of only learning an approximation to the commutator error):

$$(12) \qquad \boldsymbol{G_\theta}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) = \frac{\partial \bar{\boldsymbol{v}}}{\partial t} + \boldsymbol{m}_{\boldsymbol{\theta}}^{e}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}).$$

The symbol $\boldsymbol{m}^e$ is used to distinguish it from $\boldsymbol{m}$ because $\boldsymbol{m}^e$ generally does not approximate the commutator error. Form (12) is very generic, and many equation-discovery methods for dynamical systems fit in this form [20]. A recent example for turbulent flow simulation is [29], where the approach is called "fully learned"—no known physics are used in choosing the form of the reduced model, except that the problem is a dynamical system. Recent weather prediction models employ a similar setup [30, 31]. If $\boldsymbol{m}_{\boldsymbol{\theta}}^{e}$ is represented by a neural network, this form is similar to neural ODEs, and this will be discussed in Section 5.1.

If even less a priori knowledge is available, one can resort to so-called *surrogate models*,

$$(13) \qquad \boldsymbol{G_\theta}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) = \bar{\boldsymbol{v}}(\boldsymbol{x}, t) + \boldsymbol{m}_{\boldsymbol{\theta}}^{s}(\boldsymbol{x}, t; \boldsymbol{\mu}),$$

where $\boldsymbol{m}_{\boldsymbol{\theta}}^{s}$ represents a linear or nonlinear function approximator, such as neural networks, neural operators, polynomial expansions, Gaussian processes, dynamic mode decomposition, and proper orthogonal decomposition, taking physical parameters and spatial and temporal coordinates as input instead of the solution $\bar{\boldsymbol{v}}$. Examples of neural operators that fit in this class are the Deep Operator Network (DeepONet) [32], Fourier Neural Operator (FNO) [33], Laplace Neural Operator (LNO) [34], Graph Neural Operator (GNO) [35], and Convolutional Neural Operator (CNO) [36].

Next to the closure form (9), the generic evolution form (12), and the surrogate form (13), many other reduced forms are possible. For example, equation (11) for the reduced variable can be extended with equations for latent variables, which can be used to model the effect of memory (for details, see Section 7). In the context of learning ODEs, this leads to so-called augmented neural ODEs [37], which have been used in the context of closure models [38, 39]. In the context of the equation-free methods [40], the learned effective dynamics (LED) framework has been proposed for learning low-dimensional models that include memory [41]. Another example of a reduced form is a stochastic differential equation with trainable terms [42, 43].

---

[2] assuming the reduction operator commutes with temporal differentiation

2.4. **Example from fluid dynamics: large eddy simulation.** An important example of closure models appears in the simulation of turbulent flows with LES [17, 44]. In this case, the incompressible Navier-Stokes equations form the high-fidelity model,

$$(14) \qquad \boldsymbol{F}(\boldsymbol{u}; \boldsymbol{\mu}) := \frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) - \nabla \cdot (2\nu \boldsymbol{S}(\boldsymbol{u})) + \nabla p,$$

where $\boldsymbol{u}$ is the velocity, p the pressure, and $\boldsymbol{S}(\boldsymbol{u}) = \frac{1}{2}(\nabla \boldsymbol{u} + (\nabla \boldsymbol{u})^T)$ is the strain-rate tensor. The term $\nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u})$ represents non-linear convection (with $\boldsymbol{u} \otimes \boldsymbol{u}$ the dyadic product) and is the major cause of the closure problem. It is often written in the advective form $(\boldsymbol{u} \cdot \nabla)\boldsymbol{u}$. The equation is supplemented with the constraint $\nabla \cdot \boldsymbol{u} = 0$, but to keep the discussion concise, we omit this constraint in what follows. The reduction operator is a convolutional filter, similar to (3), which commutes with spatial and temporal differentiation so that the expression for the commutator error reads

$$(15) \qquad \boldsymbol{C}(\boldsymbol{u}, \bar{\boldsymbol{u}}) = \bar{\boldsymbol{F}}(\boldsymbol{u}; \boldsymbol{\mu}) - \boldsymbol{F}(\bar{\boldsymbol{u}}; \boldsymbol{\mu}) = \nabla \cdot (\overline{\boldsymbol{u} \otimes \boldsymbol{u}} - (\bar{\boldsymbol{u}} \otimes \bar{\boldsymbol{u}})).$$

Because the commutator error is the divergence of a quantity, the aim is to construct an approximation $\hat{\boldsymbol{m}}$ such that

$$(16) \qquad \boldsymbol{C}(\boldsymbol{u}, \bar{\boldsymbol{u}}) \approx \nabla \cdot \hat{\boldsymbol{m}}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}}; \boldsymbol{\mu}).$$

A classical model for $\hat{\boldsymbol{m}}$ is the Smagorinsky model,

$$(17) \qquad \hat{\boldsymbol{m}}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}}; \boldsymbol{\mu}) = -\nu_t(\theta) \boldsymbol{S}(\bar{\boldsymbol{u}}),$$

and the goal is to learn the Smagorinsky coefficient $\theta = C_S$, where $\nu_t(\theta) = (\theta \Delta)^2 |\bar{\boldsymbol{S}}|$ and Δ denotes the filter width [44]. The reduced model expression is in this case

$$(18) \qquad \boldsymbol{G}_{\boldsymbol{\theta}}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) := \frac{\partial \bar{\boldsymbol{v}}}{\partial t} + \nabla \cdot (\bar{\boldsymbol{v}} \otimes \bar{\boldsymbol{v}}) - \nabla \cdot (2\nu \boldsymbol{S}(\bar{\boldsymbol{v}})) + \nabla \bar{p} - \nabla \cdot (\nu_t(\theta) \boldsymbol{S}(\bar{\boldsymbol{v}})).$$

Note that even though the Reynolds-averaged Navier-Stokes equations (RANS) have a similar form, they feature a different reduction operator that leads to a much larger commutator error, and it should not be confused with LES. In many scientific machine learning approaches, instead of learning the single parameter C_S, a neural network is trained to approximate $\hat{\boldsymbol{m}}$, or $\hat{\boldsymbol{m}}$ is expressed in terms of a tensor basis with coefficients that need to be learned (more details in Section 4).

3. Objective function choice: a priori vs. a posteriori learning

Having introduced different approaches to construct a reduced model, it is clear that the resulting form involves a parameterized model with unknown parameters. This leads to the formulation of an inverse problem, where one or more parameters are learned to meet certain requirements. In this section, we continue with introducing three distinct methods for approaching the closure problem in terms of the objective function that is being minimized.

3.1. **Approach 1: A priori learning.** The most common approach in closure modeling is to find the parameters θ by minimizing an error metric that can be evaluated "offline", i.e., *without solving the reduced model* (5). This approach is known as **a priori learning**, "direct approach" [45], or "offline mode" [46]. The optimization problem has the following form:

$$(19) \qquad \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}^{\mathrm{prio}}(\bar{\boldsymbol{u}}, \boldsymbol{u}),$$

In this approach, a limited number of reference trajectories (solutions of $\boldsymbol{u}$ as a function of time for different parameters $\boldsymbol{\mu}$) are obtained from (1), from which $\bar{\boldsymbol{u}}$ can be extracted by applying the filter (3). The objective function typically involves the summation over a set of training data $\mathcal{T}$ (in parameter space),

$$(20) \qquad J_{\boldsymbol{\theta}}^{\mathrm{prio}}(\bar{\boldsymbol{u}}, \boldsymbol{u}) = \frac{1}{|\mathcal{T}|} \sum_{i \in \mathcal{T}} \mathcal{L}_{\boldsymbol{\theta}}^{\mathrm{prio}}(\bar{u}(\boldsymbol{x}, t; \boldsymbol{\mu}_i), u(\boldsymbol{x}, t; \boldsymbol{\mu}_i)),$$

where $|\mathcal{T}|$ measures the cardinality ("size") of the training data set, and $\boldsymbol{\mu}_i$ are the parameters corresponding to the i-th training sample, e.g. a vector describing initial conditions. The summation can be easily extended to include a temporal aspect. The loss function $\mathcal{L}^{\mathrm{prio}}$ can have many

different forms. A common example is to minimize the squared commutator error (see closure model form (9)):

$$(21) \qquad \mathcal{L}_{\boldsymbol{\theta}}^{\mathrm{prio}}(\bar{\boldsymbol{u}}, \boldsymbol{u}) = \|\boldsymbol{m}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}}; \boldsymbol{\mu}) - \boldsymbol{C}(\boldsymbol{u}, \bar{\boldsymbol{u}}; \boldsymbol{\mu})\|^2,$$

where the *true* commutator error $\boldsymbol{C}$ is computed according to (7) given the collected data set of $\boldsymbol{u}$ and its filtered value $\bar{\boldsymbol{u}}$. Because in this case one learns an operator, this approach is sometimes called operator fitting, or operator inference in the ROM community [47, 48] (note that this term should not be confused with the neural operators mentioned in Section 2.3). An example of such an operator is the commutator error in LES, equation (15). The general expression for the loss function is

$$(22) \qquad \boxed{\mathcal{L}_{\boldsymbol{\theta}}^{\mathrm{prio}}(\bar{\boldsymbol{u}}, \boldsymbol{u}) = \|\boldsymbol{G}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}}; \boldsymbol{\mu}) - \bar{\boldsymbol{F}}(\boldsymbol{u}; \boldsymbol{\mu}))\|^2}.$$

Figure 1 (left) shows how in a priori learning, *solving* the reduced model $\boldsymbol{G} = 0$ is not required in the optimization process; one only needs to *evaluate* the residual $\boldsymbol{G}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}}; \boldsymbol{\mu})$. Since $\bar{\boldsymbol{F}} = 0$, expression (22) effectively constitutes the residual of the approximate model $\boldsymbol{G}$, evaluated using the reference trajectory $\bar{\boldsymbol{u}}$). This means that a priori learning amounts to *residual minimization*. To evaluate the residual, the full state $\boldsymbol{u}$ is not required, but only the reduced state $\bar{\boldsymbol{u}}$.

Note that keeping $\bar{\boldsymbol{F}}(\boldsymbol{u})$ in expression (22) is insightful. Consider, for example, $\boldsymbol{F}$ and $\boldsymbol{G}$ in the form of equations (10) and (11)—the residual then reads $\boldsymbol{G}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}}; \boldsymbol{\mu}) - \bar{\boldsymbol{F}}(\boldsymbol{u}; \boldsymbol{\mu}) = \boldsymbol{m}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}}; \boldsymbol{\mu}) - (\bar{\boldsymbol{f}}(\boldsymbol{u}; \boldsymbol{\mu}) - \boldsymbol{f}(\bar{\boldsymbol{u}}; \boldsymbol{\mu}))$, which shows that $\boldsymbol{m}$ models a commutator error.

A priori learning with operator fitting is a common approach in the closure modeling community, and many examples are available: learning stress tensors (or divergence of stress tensor) in fluid dynamics [49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61], learning a polymer stress tensor or the potential function in molecular dynamics [62], learning closure terms by representing them using a neural operator [63], and parameterization in climate models [64].

The crucial aspect of a priori learning is that evaluating the objective function (19) only requires reference data $\boldsymbol{u}$ and $\bar{\boldsymbol{u}}$, obtained as solution vector from solving the high-fidelity model. It involves the accuracy of the operator, but not the accuracy of the solution $\bar{\boldsymbol{v}}$ that will be computed when the learned operator is employed (which is known as the "online" mode [46], or the "a posteriori" analysis). A common problem is that even with a highly accurate operator fit, the solution $\bar{\boldsymbol{v}}$ can *drift* from the true solution $\bar{\boldsymbol{u}}$ or becomes *unstable*. The issue is also known under the term *model-data inconsistency* [65, 66], where the data used to train the closure models is not consistent with the environment in which the model will be run, e.g., due to discretization effects.

In turbulence, the instability is often associated with the concept of backscatter: energy transfer from the small scales back to the large scales [67]. For example, [58] reported unstable results and applied clipping to limit backscatter. Beck et al. [55] performed a projection of their neural-network closure model onto an eddy viscosity basis to enforce stability, and Kurz et al. [66] performed "stability training" by adding noise to the training data. This approach of adding noise is common in the machine learning communities, see for example [29] and [68]. Rasp et al. [46] proposed coupled online learning in which the (offline-trained) machine learning model is further corrected by nudging with a high-fidelity model that is run in parallel. Pawar and San used data assimilation techniques to correct the trained model in an online environment [69]. Yuval et al. [70] address the stability issue by employing random forests, which automatically respect energy conservation. Charalampopoulos et al. [71] used imitation learning to stabilize their results. Guan et al. [67] showed that increasing the size of the training set can give stable results. Jakhar et al. [72] used a sparsity-promoting linear regression technique to construct interpretable closed-form equations. A more general example of using a-prior closures on dynamical systems was presented in [73] where it was observed that such techniques did not reproduce extreme events from the original system. Pedersen et al. [74] proposed to add a term into the a priori loss function that measures the mismatch between the true reduced model state and a prediction by another neural network (called an emulator). By using this emulator instead of the actual reduced model, it is possible to include the solution error into the loss function without needing to resort to a posteriori learning. Similar ideas (i.e., using high-fidelity data to inform parameterizations in low-order models) are also observed in other multiscale modeling applications such as for the mechanical analysis of materials [75, 76].

In summary, the main advantages of the a priori approach are the relative ease of training (no differentiable solvers needed) and the prospect of generalization and interpretability, i.e., the possibility to reuse the learned operator when geometry, boundary conditions, or even the differential equation itself changes. However, the main disadvantage is that the solution of the reduced model is not part of the error metric, so that instability and drift can lead to inaccurate solutions. We note that in the reduced order modeling literature, conditions have been derived under which non-intrusive models recover the same models that would be obtained with intrusive projection-based model reduction [48]. Similar techniques might be useful to derive conditions under which the learned operators in a-priori learning can be reused.
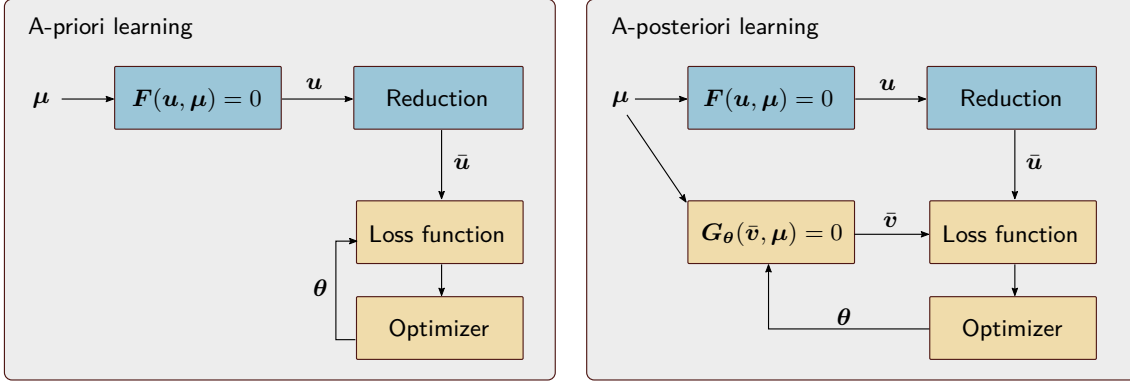
FIGURE 1. A priori vs. a posteriori learning. Training data $\boldsymbol{u}$ and $\bar{\boldsymbol{u}}$ are obtained from solving the high-fidelity model $\boldsymbol{F}$. In blue: simulation of the high-fidelity model, from which the ground truth $\bar{\boldsymbol{u}}$ is derived. In yellow: training. In a posteriori learning, the loss function (24) includes evaluating the gradients of the reduced model, while in a priori learning, only the residual must be evaluated (equation (22)).

3.2. **Approach 2: A posteriori learning.** An alternative approach that addresses some of the issues of a priori learning is to minimize an error metric that involves querying the model being learned itself to compute $\bar{\boldsymbol{v}}$. This amounts to solving the following minimization problem

$$(23) \qquad \boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}^{\mathrm{post}}(\bar{\boldsymbol{v}}, \bar{\boldsymbol{u}}),$$

where J^{post} is a suitable objective function, and $\bar{\boldsymbol{v}}$ satisfies the reduced model equation (5). If the reduced model equation is a PDE, then this is effectively a PDE-constrained optimization problem or an optimal control formulation, as will be detailed in Section 6.2. Compared to a priori learning, the difference lies in the arguments: $\bar{\boldsymbol{u}}$ and $\bar{\boldsymbol{v}}$, instead of $\boldsymbol{u}$ and $\bar{\boldsymbol{u}}$ in (19). This approach is called **a posteriori learning** because it involves solving the reduced model equation—see Figure 1 (right).

The objective function J^{post} again involves summation of a loss function over a set of training data. In addition to the possibility of residual minimization (22), a posteriori learning offers a second possibility, namely solution error minimization:

$$(24) \qquad \mathcal{L}_{\boldsymbol{\theta}}^{\mathrm{post}}(\bar{\boldsymbol{v}}, \bar{\boldsymbol{u}}) = \begin{cases} \|\boldsymbol{G}_{\boldsymbol{\theta}}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) - \bar{\boldsymbol{F}}(\boldsymbol{u}; \boldsymbol{\mu}))\|^2, & \text{residual minimization,} \\ \|\bar{\boldsymbol{v}}_{\boldsymbol{\theta}} - \bar{\boldsymbol{u}}\|^2, & \text{solution error minimization.} \end{cases}$$

These two loss functions can be used individually, but can also be combined into a single loss function with a weighting factor. An example of residual minimization is again the commutator loss $\|\boldsymbol{m}_{\boldsymbol{\theta}}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) - \boldsymbol{C}(\boldsymbol{u}, \bar{\boldsymbol{u}}; \boldsymbol{\mu})\|^2$, but now with $\boldsymbol{G}$ depending on $\bar{\boldsymbol{v}}$, not $\bar{\boldsymbol{u}}$ as in (22). An example of solution error minimization is the integral loss $\int_0^T \|\bar{\boldsymbol{v}}_{\boldsymbol{\theta}}(\cdot, t) - \bar{\boldsymbol{u}}(\cdot, t)\|^2 \mathrm{d}t$, which is sometimes known as "trajectory fitting" [77, 78, 79]. We note that pure dependence on $\bar{\boldsymbol{u}}$, and not on $\boldsymbol{u}$, as suggested in (23) is not always feasible, and the residual minimization in (24) is an example of this.

Again, compared to a priori learning (19), $\mathcal{L}^{\mathrm{post}}$ in (24) is also a function of $\bar{\boldsymbol{v}}$ (which depends on $\boldsymbol{\theta}$), whereas $\mathcal{L}^{\mathrm{prio}}$ was only a function of the reference data $\boldsymbol{u}$ and $\bar{\boldsymbol{u}}$, *which do not depend on $\boldsymbol{\theta}$.* This makes the optimization problem more difficult to solve, and differentiable solvers are typically

needed. For example, in order to compute $\frac{\partial J}{\partial \boldsymbol{\theta}}$, the Jacobian $\frac{\partial \bar{\boldsymbol{v}}}{\partial \boldsymbol{\theta}}$ is needed—this requires that the code that solves $\boldsymbol{G}_{\boldsymbol{\theta}} = 0$ is differentiable, or that an adjoint solver is available. In addition to a posteriori learning, the approach is known under various other names, such as "solver-in-the-loop" [80], "curriculum training" [81], "indirect approach" [45], "end-to-end learning", "differentiable physics", and "online learning" [7].

The a posteriori learning framework is not limited to either residual or solution error minimization; one could also combine these two errors into a hybrid loss function, or use quantities of interest like energy instead of the solution itself. For example, List & Thuerey [80] propose a loss function composed of several terms, and Beck & Kurz [82] employ the energy spectrum in the loss function. Such hybrid methods have the potential to bridge the common discrepancy between functional and structural approaches (see, e.g., [83]).

With $\boldsymbol{G}$ in closure model form (9), the residual minimization of the a posteriori formulation (24) resembles the a priori formulation (22). In contrast to a priori learning, the a posteriori approach aims to learn the term $\boldsymbol{m}_{\boldsymbol{\theta}}(\bar{\boldsymbol{v}}; \boldsymbol{\mu})$ *such that the solution $\bar{\boldsymbol{v}}$ obtained when solving the reduced model* (24) *is close to the reference trajectory $\bar{\boldsymbol{u}}$.*

A posteriori learning is a relatively recent technique in the closure modeling field, and less literature has appeared in comparison to a priori learning[3]. Early examples of a posteriori training of neural networks in PDEs (mainly fluid dynamics) are the direct field-inversion approach of Duraisamy and collaborators [85] (in the context of RANS), and the Deep learning PDE Model (DPM) of Sirignano et al. [86] (in the context of LES), which solves the adjoint equations to compute the gradients of the loss function (also known as "sensitivities"). In [45], the DPM is extended to train on mean statistics instead of the solution field itself, and efforts are made to interpret the learned neural network closure, as well as its dependence on discretization and the grid. Similarly, [80] learns the sub-grid-scale stress tensor, but uses automatic differentiation to obtain gradients and proposes to use a loss function that contains several terms, reflecting errors in the solution, in the stress tensor, and in the energy spectrum. In [87], several reduced model forms are considered, ranging from neural ODEs (ResNets) to models that learn a single parameter (the Smagorinsky coefficient); this was extended in [88] where a neural network model is used as additive correction to the learned Smagorinsky model. In [89], a posteriori learning is used to learn the stress tensor in quasi-geostrophic turbulence, which is a particularly challenging application because it exhibits a strong energy transfer from small to large scales.

An ongoing theme in a posteriori learning is how many time steps are "unrolled" in evaluating and back-propagating the loss function. Unrolling too few time steps gives only limited gains over a priori learning, while unrolling too many time steps is computationally expensive, has the danger of exploding or vanishing gradients, and can be unrealistic given that turbulent flows are chaotic, meaning that initially close trajectories are expected to diverge [77, 90, 91]. One strategy is to unroll predictions for a small number of the time steps and then only back-propagate to the last time step; this is called the pushforward method [92]. Recently, a similar a posteriori learning approach has appeared in the ROM literature, in which similar challenges are faced [93, 94].

In summary, the main advantage of the a posteriori approach (with solution error minimization) is that one directly targets the accurate approximation of $\bar{\boldsymbol{u}}$, which is typically the quantity one is interested in. This has been shown to improve stability compared to a priori learning. In addition, if the form of $\boldsymbol{G}$ does not involve $\boldsymbol{F}$, like in the evolution form (10), access to $\boldsymbol{F}$ is not required, and only reduced solution trajectories $\bar{\boldsymbol{u}}$ are needed. The approach implicitly corrects for spatial and/or temporal discretization errors, which can be desirable but can also limit application to different grids or time steps (further discussed in Section 5). The main disadvantage of the a posteriori approach is its computational expense, and depending on the precise form of the reduced model, it requires differentiable solvers (typically using adjoints or automatic differentiation) and a judicious choice of the number of unrolled time steps in the loss function.

3.3. Approach 3: Reconstruction.
A third approach, which can be used in both a priori or a posteriori fashion, is the reconstruction of the full solution field $\boldsymbol{u}$ (given the reduced field $\bar{\boldsymbol{u}}$) and then applying the original model $\boldsymbol{F}$. So, instead of trying to minimize the error in the *reduced*

---

[3]note that in PDE-constrained optimization this approach has been in use for much longer [84]

solution (as given by (24)), the goal is to minimize the error in the *reconstructed* solution:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} J_{\boldsymbol{\theta}}^{\mathrm{r}}(\bar{\boldsymbol{u}}, \boldsymbol{u}) \tag{25}$$

where the loss function typically involves a solution error

$$J_{\boldsymbol{\theta}}^{\mathrm{r}}(\bar{\boldsymbol{u}}, \boldsymbol{u}) = \|\mathcal{R}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}}) - \boldsymbol{u}\|^2, \tag{26}$$

or a residual error, e.g.,

$$J_{\boldsymbol{\theta}}^{\mathrm{r}}(\bar{\boldsymbol{u}}, \boldsymbol{u}) = \|\bar{\boldsymbol{F}}(\mathcal{R}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}})) - \bar{\boldsymbol{F}}(\boldsymbol{u})\| = \|\frac{\partial \bar{\boldsymbol{u}}}{\partial t} + \bar{\boldsymbol{f}}(\mathcal{R}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}}))\|^2, \tag{27}$$

and the goal is to find the parameters of the reconstruction operator $\mathcal{R}_{\boldsymbol{\theta}}$. Once $\boldsymbol{\theta}$ is determined, one solves

$$\bar{\boldsymbol{F}}(\mathcal{R}_{\boldsymbol{\theta}}(\bar{\boldsymbol{v}}); \boldsymbol{\mu}) = 0. \tag{28}$$

We call this approach **reconstruction**, although the term deconvolution is also commonly used [44, 95]. The main challenge is obviously to determine $\mathcal{R}$, which is an inverse problem that is generally ill-posed. In addition, the reconstruction operator is a non-local operator [96]. Classically, the reconstruction operator has been determined using approaches like the iterative van Cittert method [17]. Recently, machine learning alternatives have been proposed, in which $\mathcal{R}$ is modeled by a neural network [97, 98, 99, 100, 101]. Neural networks are promising because identifying the reconstruction operator is an ill-posed problem that can benefit from data-driven approaches. These methods can be considered an improvement over approximate deconvolution techniques because they do not need the assumption of an invertible filter, but open challenges include how such data-driven methods can generalize across physical regimes, numerical schemes, and discretizations.

With objective function (25), reconstruction is a special case of a priori learning. By replacing $\bar{\boldsymbol{u}}$ with $\bar{\boldsymbol{v}}$ in (25) and (26), the approach can also be formulated in terms of a posteriori learning, but this is not common in literature [97, 98, 99, 100]. Compared to formulation (23), there is less guarantee that the solution $\bar{\boldsymbol{v}}$ will be close to $\bar{\boldsymbol{u}}$. Additionally, the use of a posteriori reconstruction operator learning is limiting because of dramatic increases in memory requirements to reconstruct the full flow-field as an indirect route to computing subgrid stresses. We note that the objective functions (26) and (27) are similar to those used when training autoencoders [102].

3.4. **From offline to online learning.** The a priori and a posteriori learning approaches from Sections 3.1 and 3.2 are generally used in an offline setting, where the term "offline" is used here to indicate that the high-fidelity model is not required in the learning phase. Once training data has been generated by running the high-fidelity model at different parameter values, the high-fidelity model is not needed anymore, and the reduced model can be learned (in a priori or a posteriori fashion). In contrast, recent research is being directed at *online* learning, in which one includes the high-fidelity model in the learning phase. For example, Rasp et al. [46] propose to run the high-fidelity model in parallel with the training process so that it can nudge the reduced model into the right direction and prevent instabilities. Eventually, the idea is that the trained model becomes accurate and stable enough so that it can be run without this coupling. Another example is the dynamic deep learning closure method [103], in which high-fidelity simulations are run in parallel with the reduced model, but only on a small part of the simulation domain. The closure model parameters are dynamically adjusted based on the difference between the high-fidelity model and reduced model on this subdomain, and then used throughout the entire domain. With this approach, no DNS training data needs to be stored, and there is no issue with extrapolation in time or in parameters because the data used for learning the closure model is fit for a purpose.

We note that the concept of online learning is closely related to the setting in which reduced model parameters (and state) are continuously updated through (observation) data that arrives sequentially in time—we refer to this as data assimilation (see Section 6.3). An example is a neural network-based closure model that is enhanced by using data assimilation techniques [69]. In addition, another related concept is that of *active* learning [104, 105], which is typically used to indicate that the samples of the high-fidelity model are iteratively chosen, based for example on the accuracy of the reduced model—see [106, 107].

3.5. **Reinforcement learning.** The objective functions in Sections 3.1-3.3 fit within the framework of supervised learning: the high-fidelity model provides labeled data, and the purpose of the reduced model is to reproduce this accurately. Some limitations of the supervised approach are the need for generating training data (expensive), the issue of defining the reduction operator and reduced training data (the reduction operator is, in practice, often implied by the discretization and therefore not fully known—see Section 5), and the issue of stability, robustness, and the possible need for differentiable programs (see Section 3.1). Reinforcement learning promises to circumvent such issues [108, 109, 60, 110, 111]: it trains an agent while it is interacting with a dynamical environment. This agent is typically a neural network and the environment is the reduced model, and the agent acts to determine a policy (e.g., the Smagorinsky coefficient, see 2.4) such that a certain reward (objective function) is optimized. For recent review papers, see [112, 16]. Some listed advantages of the approach are that only statistics and no high-fidelity data are required, it does not employ adjoints or differentiable solvers (but approximates gradients of the cost function through sampling), and it works directly in the fully discrete reduced model to account for both the closure model error as well as for discretization errors [109]. Compared to earlier sections, reinforcement learning shares most similarity with a posteriori learning. However, one should realize that with a posteriori learning, training based on statistics is also possible, as well as accounting for the discretization error, and gradients can be computed more accurately.

4. Physics-constrained learning

The reduced model forms for G discussed in Section 2 differed in the degree to which "known physics" was included. Depending on the application, it is often possible to further inject physics knowledge into the form of G or in the objective function. In general, the ability to respect physical laws such as conservation and invariance are of key importance for the broad-spectrum utilization of machine learning models for computational physics applications. The development of data-driven models that adhere to such laws leads to improved interpretability and explainability, generalization and robustness, and data efficiency [28, 113]. Therefore, there is a concerted effort to develop algorithms that have constraints embedded in them to avoid violating physical laws. Constraints can be embedded in such data-driven models in a "soft" manner where data augmentations or weak penalties in loss functions are used to approximately satisfy constraints. A popular example of such a model is a PINN, mentioned in Section 2, that penalizes deviations from governing laws as a residual term in the model optimization [28, 27, 24]. These models are typically easier to construct and can be readily applied to existing workflows. In contrast, "hard" constraints require the construction of machine learning models that can satisfy constraints by design. While these algorithms require greater turnaround time for construction, training, and deployment, they can guarantee constraint satisfaction during interpolation as well as extrapolation. Examples of model development that constrain a data-driven model by design to permitted behavior include symmetry preservation in graph neural networks (such as for molecular dynamics [62] and rotationally invariant flows [114]). Figure 2 shows an example of imposing mass conservation for a 2D incompressible flow as a soft constraint (in a PINN-like fashion), compared to hard constraint (through the definition of streamfunction). Although the hard constraint approach satisfies the divergence-free condition by construction (both during training and deployment), the prediction of the streamfunction is a non-trivial task by itself. In the following sections, we explore certain examples of soft and hard constraint-based machine learning with applications to closure modeling.

4.1. **Soft constraints.** The easy application of soft constraints to machine learning models makes their use attractive for several turbulence closure modeling strategies. In the scientific machine learning community, PINNs [24] are probably the most popular approach for training neural networks using soft constraints, originally introduced by Lagaris et al. [115]. These constraints are typically in the form of loss-function regularizations that comprise governing laws. Eivazi et al., [116], proposed the use of PINNs for solving the Reynolds-averaged Navier-Stokes equations. In this study, the authors were able to bypass the use of specific model forms for the Reynolds stresses by simply using a governing equation residual loss. A similar study is seen in [117], where sparse experimental data is also assimilated in the process of training a PINN. Ref. [118] introduces a
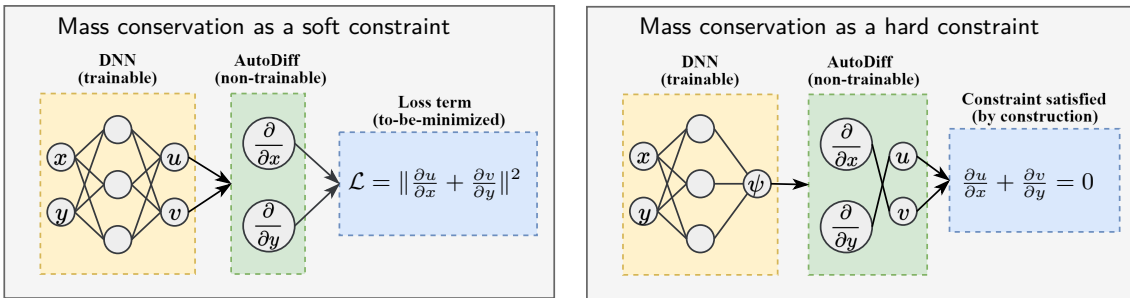
FIGURE 2. An example of imposing mass conservation for a 2D incompressible flow as a soft constraint (left) and hard constraint (right). For the hard constraint, the streamfunction ψ is learned instead of the velocity field; the velocity field follows as $\boldsymbol{u} = \nabla \times \psi$, which by construction satisfies $\nabla \cdot \boldsymbol{u} = 0$.

two-model formulation where PINN models are trained for low- and high-wavenumber information to obtain more accurate predictions for two-dimensional turbulence. It was observed that a dedicated architecture for high-wavenumber information aided in generalization, circumventing the spectral bias problem [119]. In [61], a hybrid formulation is used where a closure is constructed with enstrophy conservation enforced by a loss-term regularization and where rotational equivariance is preserved by a group-equivariant convolutional neural network. The formulation is applied to construct a closure for a two-dimensional turbulent flow. In [120], a loss-function augmentation is used to restrict the evolution of a machine learning closed system to a low-dimensional phase space to avoid a posteriori errors. This augmentation improved the quality of an a priori trained closure without increased propagating error during a posteriori deployment. Finally, another approach to satisfying a class of constraints weakly is through the use of data augmentation. Here, augmentations to training datasets can be performed to implicitly inform the learning algorithm about invariances or equivariances in the function approximation. For example in [121], data augmentation with Lie symmetries not only leads to the improved satisfaction of the same but also improves sample complexity. Similar operations were also deployed in [122] for generating synthetic turbulence snapshots. In [123], a similar data augmentation was performed for learning subgrid stress tensors in two-dimensional turbulence with improved results compared to the standard dataset.

4.2. **Hard constraints.** Hard constraints are typically application specific, and we focus on turbulence closure modeling. It is well known that subgrid stress tensors in LES, or Reynolds stress tensors in RANS, obey certain laws such as aggregate dissipative behavior, symmetry, invariance, etc. [124, 125]. Therefore, it is desirable to construct data-driven prediction techniques for these quantities that adhere to these constraints during inference. The simplest efforts to constrain predictions from data-driven closures have involved ad-hoc truncations that preserve numerical stability in deployments. These typically occur via clipping to ensure guaranteed dissipation of the subgrid stress [59, 55]. Other efforts include constrained predictions via projection of data-driven predictions onto the so-called realizability triangle [126]. Another popular approach for constructing data-driven approximations of such tensors are via tensor-basis neural networks (TBNNs) [50, 51, 126, 127], which guarantee a Reynolds stress prediction as a linear expansion of strain and rotation tensors following Pope's generalized eddy viscosity hypothesis [128]. The stress tensor $\boldsymbol{\tau}$ - denoted by $\hat{\boldsymbol{m}}_{\boldsymbol{\theta}}(\bar{\boldsymbol{u}}; \boldsymbol{\mu})$ in (16) - is expressed as

$$(29) \qquad \boldsymbol{\tau} = \sum_{i=0}^{10} \alpha_i \boldsymbol{T}_i,$$

where the coefficients α_i are functions of a set of scalar invariants $\lambda_1, \ldots, \lambda_5$, which in turn depend on the strain-rate tensor $\boldsymbol{S}$ and the rotation tensor $\boldsymbol{R} := \frac{1}{2}(\nabla \boldsymbol{u} - (\nabla \boldsymbol{u})^T)$, and $\boldsymbol{T}$ is the tensor integrity basis, also a function of $\boldsymbol{S}$ and $\boldsymbol{R}$. For example, $\lambda_1 = \text{tr}(\boldsymbol{S}^2)$ and $\boldsymbol{T}_1 = \boldsymbol{S}$. The functions $\alpha_i = f_i(\lambda_1, \ldots, \lambda_5; \boldsymbol{\theta})$ are modelled as neural networks with parameters $\boldsymbol{\theta}$. In the large eddy simulation context, similar symmetries may also be leveraged for constrained predictions from neural network models [125, 129, 124, 130]. The important message is, that by choosing the form

(29), one has directly encoded the Galilean, rotational, and reflectional invariances of the stress tensor, independent of the neural network architecture or its parameters.

In several applications, property preservation can be posed as linear constraints. In such cases, neural network architectures that guarantee satisfaction of these constraints have also been proposed [131, 132]. These include the solution of an additional system of equations to project the output of the neural network to a subspace that satisfies these equations. However, it must be noted that this involves the inversion of a matrix that grows with the size of the output for each forward pass of the neural network. Another approach pioneered in [39] introduces an additional set of latent variables and specifies an explicit form of the closure (a combination of skew-symmetric and dissipative terms) to guarantee energy conservation and stability of the closure; see also [71] for related work. In reduced-order models, similar ideas have been used [133, 3, 134].

Finally, a popular approach to building closure models in data-limited scenarios is through field inversion and machine learning [135], which allows for the PDE-constrained optimization of defects in turbulence models using experimental or sparse sampling data. If the defects to turbulence models are appropriately defined, constraints can be satisfied by design both during the optimization as well as the deployment procedure [136]. A similar procedure may also be followed via reinforcement learning for closure modeling [111, 108, 137] (see also Section 3.5), which offers the advantage of sparse data requirements, although computational costs associated with learning are significant in the absence of gradient information. We would like to emphasize that hard and soft constraints are not mutually exclusive and that data-driven modeling approaches can blend both concepts for balancing accuracy, constraint satisfaction, and ease of learning and deployment. For example, in [138], hard constraints are used for tensor symmetries using TBNNs and soft constraints are imposed for the smoothness of heat fluxes.

In the more general context of symmetry preservation for fluid flow forecasting, several novel neural network algorithms have been proposed that can be repurposed for closure modeling. Wang et al. [139] incorporate translational, scaling, and rotational equivariance of the Navier-Stokes equations in the neural network design. In [140, 141], novel convolutional neural network architectures that preserve frame invariance are used for fluid-flow predictions. Specifically, Pawar et al. [141] use a novel architecture for improved subgrid stress modeling of two-dimensional large eddy simulation. Guan et al. [61] used group convolutional neural networks and demonstrated that the preservation of symmetries alleviated the need for much larger datasets.

Another approach to preserving desirable properties during regression tasks is via the use of non-parametric methods like random forests. In [70], a random-forest regressor is used for predicting subgrid stresses for climate models, with guaranteed satisfaction of symmetry preservation. This is because random-forest predictions are obtained by averaging splits of data that come from the convex hull of the training data set. As long as the generated training data sets guarantee a particular symmetry or structure, the prediction of the dependent variable will also satisfy this property by design. Note that for complicated properties that emerge from complex interactions between machine learning predictions and the numerical solver, such an approach may be incomplete.

5. DISCRETIZATION ASPECTS

The discussion in Sections 2 - 4 was mostly agnostic of the spatial and temporal discretization methods needed to solve the true model (1) and approximate model (5). Spatial and temporal discretizations give rise to additional levels of approximation, and we indicate by $\boldsymbol{F}_h$ the (fully discrete) *high-fidelity model*, which satisfies

$$(30) \qquad \boldsymbol{F}_h(\boldsymbol{u}_h^n; \boldsymbol{\mu}) = 0,$$

where $\boldsymbol{u}_h^n \approx \boldsymbol{u}(\boldsymbol{x}, t^n)$. Similarly, $\boldsymbol{G}_{h,\boldsymbol{\theta}}$ denotes the (fully discrete) *low-fidelity model*, which satisfies

$$(31) \qquad \boldsymbol{G}_{h,\boldsymbol{\theta}}(\bar{\boldsymbol{v}}_h^n; \boldsymbol{\mu}) = 0,$$

and $\bar{\boldsymbol{v}}_h^n \approx \bar{\boldsymbol{v}}(\boldsymbol{x}, t^n)$.

5.1. **Spatial discretization and neural ODEs.** In practice, the low-fidelity model is often constructed based on a high-fidelity model that is only spatially discretized, leaving time continuous. An important example is when the high-fidelity model is in the form of equation (10), leading to

$$(32) \qquad \boldsymbol{G}_{h,\boldsymbol{\theta}}(\bar{\boldsymbol{v}}_h; \boldsymbol{\mu}) = \frac{\mathrm{d}\bar{\boldsymbol{v}}_h}{\mathrm{d}t} + \boldsymbol{f}_h(\bar{\boldsymbol{v}}_h; \boldsymbol{\mu}) + \boldsymbol{m}_{h,\boldsymbol{\theta}}(\bar{\boldsymbol{v}}_h; \boldsymbol{\mu}).$$

In this case, the spatial discretization $\boldsymbol{f}_h$ of the high-fidelity model can be reused but on a coarser grid, and existing high-fidelity codes can simply be extended by adding a closure term. Instead of a coarser grid, a low-order method can also be used, like in [142]. Furthermore, this notation also covers the case where $\bar{\boldsymbol{v}}_h$ is being represented by the coefficients of an ROM, see, e.g., [143, 3].

In scientific machine learning, the common choice is to represent $\boldsymbol{m}_{h,\boldsymbol{\theta}}$ by a neural network, resulting in so-called neural (closure) ODE results [143, 77, 144, 145] (the concept has been denoted as the Universal Differential Equation in [146]). Casting the closure problem as a neural ODE has the benefit that one can formulate the neural network training as an optimal control problem, which can be solved for example by using adjoint methods, as proposed in [144], and the parameters of the network can be interpreted as the control variables. One difference is that in the classical optimal control setting [147, 148] the control variables are time-dependent, whereas the neural network parameters in [144] are time-independent. These are important differences, some of which are addressed in [149]. A recent approach in which the closure model parameters change dynamically over time is given in [103].

Using neural networks for closure modeling of form (32) is advantageous because time is continuous, so the neural networks are expected to also work when different time integrators or different time steps are used. However, the fact that one is dealing with a spatially discretized system means that the neural network is linked to the particular discretization method and grid that were present in the training data, and will not easily generalize to different grids or different discretization methods. On the other hand, an advantage of learning discretized operators is that they can correct for discretization errors [86].

As mentioned, the time-continuous nature of neural ODEs allows one to formulate the adjoint equations associated with the optimization problem (24), see, e.g., [144], which can be solved backward in time to yield the sensitivities ($\frac{\partial \mathcal{L}}{\partial \theta}$). Alternatively, one could formulate the time-discrete equations and then determine sensitivities by backpropagating through the entire ODE solver. Optimize-then-discretize or discretize-then-optimize is a topic that has been discussed in the optimization literature—see textbooks like [84, 150]. Figure 3 shows a pictorial example of both paradigms. Recent research suggests that there are some important benefits to discretize-then-optimize [151, 77], like obtaining the exact gradient (of the time-approximate solution) instead of the approximate gradient that is returned by adjoint methods.
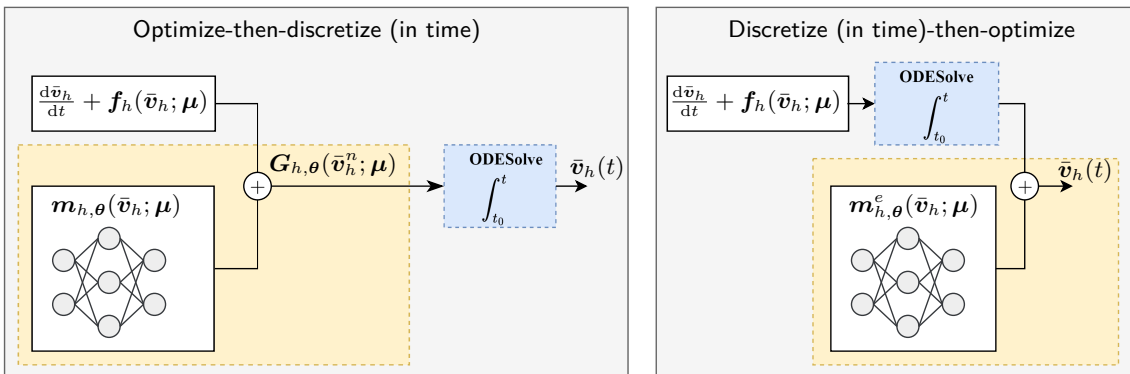

FIGURE 3. Optimize-then-discretize vs. discretize-then-optimize approaches. In orange: the optimization of a neural network to learn a time-continuous closure (left) or a time-discrete correction (right). In blue: the time integration/discretization step.

5.2. **Temporal discretization and autoregressive methods.** An example of a "discretize-then-optimize" formulation is

$$(33) \qquad \bar{\boldsymbol{v}}_h^{n+1} - \bar{\boldsymbol{v}}_h^n + \Delta t \left(\boldsymbol{f}_h(\bar{\boldsymbol{v}}_h^n; \boldsymbol{\mu}) + \boldsymbol{m}_{h,\boldsymbol{\theta}}^e(\bar{\boldsymbol{v}}_h^n; \boldsymbol{\mu}) \right) = 0,$$

which corresponds to (31) when the discretized evolution form is taken for $\boldsymbol{G}_{h,\boldsymbol{\theta}}$. Although there are several advantages of discrete methods (exact gradients, correction for discretization errors),

one important disadvantage is that the closure model is not only linked to the spatial discretization but also to the time step used during training.

Beyond the closure forms (9) and (11), fully discrete approaches are quite commonly used in reduced models of the evolution form (12), in particular so-called autoregressive methods. Although these fall strictly speaking beyond the concept of closure modeling, they deserve to be mentioned given their recent success in weather prediction [30, 31] and in general in their use in predicting chaotic dynamical systems [152, 41]. The main idea is to generalize the discrete version of (12), e.g., $\bar{\boldsymbol{v}}_h^{n+1} = \bar{\boldsymbol{v}}_h^n - \Delta t \boldsymbol{m}^e(\bar{\boldsymbol{v}}_h^n; \boldsymbol{\mu})$, to the form

$$(34) \qquad \bar{\boldsymbol{v}}_h^{n+1} = \mathcal{N}_{\boldsymbol{\theta}}(\bar{\boldsymbol{v}}_h^n, \bar{\boldsymbol{v}}_h^{n-1}, \bar{\boldsymbol{v}}_h^{n-2}, \ldots; \boldsymbol{\mu}).$$

In other words, one learns an operator $\mathcal{N}_{\boldsymbol{\theta}}$ that uses past solution states to perform time stepping. Even though the closure form (33) intuitively seems easier to learn because it involves only a "correction" to the known physics, its mathematical form is less expressive, and it has been observed that this limits the model performance [153]. In fact, (34) has a non-Markovian form, allowing it to incorporate history, which has its foundations in the Mori-Zwanzig theory, as will be discussed in Section 7. The loss function typically involves the error in the solution, like in Section 3.2, so this is a type of a posteriori learning. Given that multiple past states are included, the computation of the derivatives of the loss function through back-propagation will be more involved than for a Markovian approach like (33). In addition, one would expect again that the learned operator is only accurate for particular time steps. However, the impressive performance of weather predictions [30, 31, 154] indicates that this is perhaps less of a concern for practical problems. An example of an autoregressive method for LES is given in [155], where an FNO is employed. Autoregressive methods have also been used in message-passing PDE solvers [156].

Another example where form (34) is employed, is in 'model-free' machine learning for forecasting of chaotic systems [157, 152, 41]. In that case, $\bar{\boldsymbol{v}}_h$ typically consists of an internal hidden memory state and an observable, which are evolved in time with a recurrent neural network [41]. The foundation behind this approach lies in Takens' theorem [158], which states that a dynamical system can be identified from partial observations of the state, as long as sufficient history (delays) of these partial observations is available. This is reminiscent of the Mori-Zwanzig formalism, which offers a framework to reduce the dimension a dynamical system by including history effects. However, Takens' theorem instead aims at reconstructing the full state based on time-delayed observations.

5.3. **Learned discretizations.** An approach that is related to the two previous sections entails directly learning a (coarse) discretization scheme that implicitly includes the effect of the closure term, such as the "learned interpolation" method [159]. So, instead of employing the spatial discretization of the high-fidelity model on a coarse mesh and adding a correction term to it, cf. (32), one learns a discretization scheme (an improved interpolation method based on neural networks) for a coarse grid that effectively takes into account unresolved physics. This approach is rather intrusive because it requires one to change an existing code at the level of the discretization, but comes with the advantage that the learned interpolation (effective discretization stencil) is in some sense interpretable. Interestingly, the implicit LES formulation [160] (which was developed long before machine learning came into play) also falls within this category—it does not feature a closure model, but relies on the fact that the discretization typically introduces additional dissipation compared to the true physics, which mimics the additional dissipation that is usually provided by the closure model.

A less intrusive approach is the "learned correction" approach [161, 153, 162], which is simpler to implement and more flexible. After performing a time step with the coarse grid operator, the solution is corrected with a learned term. This approach is in fact similar to the fully discrete formulation (33) when the term $\Delta t \boldsymbol{m}_{h,\boldsymbol{\theta}}^e$ is identified as a "learned correction". In [29], a similar approach is taken, but the known physics part expressed by $\boldsymbol{f}_h$ is not included, so the neural network has to learn the entire effect of both known physics and correction.

It should be noted that in many practical LES models, the filtering operation (3) is not explicitly defined, but implied by the discretization [111]. The alternative, namely explicit filtering, has the important advantage that it converges to the "true filtered" equations upon grid refinement [163, 164]. However, because this is generally considered to be computationally too expensive, the

implied filter approach is often preferred. The disadvantage, however, is that it is difficult to define filtered reference data (needed to train neural networks) because the filter is not explicitly known.

5.4. Discretization invariance.

All approaches that are spatially or temporally discrete suffer to some extent from the issue that generalization to different grids, time steps, and discretization methods is difficult. Neural operators like DeepONet and FNO are an important step in the direction of methods that are grid-independent because they are mappings between (infinite-dimensional) function spaces [33, 165, 32, 166]. In practice, training data for such operators still involves discrete input and output data. Recently, a class of neural operators has been proposed in which the discrete input-output representation and their continuous function space realizations are equivalent [167].

6. CONNECTIONS TO INVERSE PROBLEMS, PDE-CONSTRAINED OPTIMIZATION, AND STATISTICAL METHODS

6.1. Inverse problem specification.

The minimization problems (19), (23) and (25) from Section 3 can be interpreted as inverse problems. They are similar in that some output quantity is known based on the high-fidelity simulation (e.g., the exact reduced solution or the exact reduced operator), and one can try to find a model that best approximates that output. The inverse-problem nature implies that a unique solution typically does not exist and that solutions can be sensitive to perturbations.

For example, the a posteriori learning problem can be formulated in terms of an inverse problem, which typically features two components: a forward model,

$$(35) \qquad \boldsymbol{G_\theta}(\bar{\boldsymbol{v}};\boldsymbol{\mu}) = 0,$$

and an observation equation,

$$(36) \qquad \boldsymbol{y} = \boldsymbol{h_\theta}(\bar{\boldsymbol{v}}) + \varepsilon,$$

where $\boldsymbol{y}$ denotes the observations and $\boldsymbol{h}$ denotes the observation operator. In inverse problem terms, the art is to find the parameters $\boldsymbol{\theta}$ given the observations and the forward model. In the closure model framework outlined in Sections 2 and 3, the observations come from high-fidelity simulations and the observation operator extracts the corresponding data from the reduced model. For example:

$$(37) \qquad \text{observations of the state:} \qquad \boldsymbol{y} = \bar{\boldsymbol{u}}, \qquad \boldsymbol{h_\theta}(\bar{\boldsymbol{v}}) = \bar{\boldsymbol{v}},$$

$$(38) \qquad \text{observations of the closure term:} \qquad \boldsymbol{y} = \boldsymbol{C}(\boldsymbol{u},\bar{\boldsymbol{u}};\boldsymbol{\mu}), \qquad \boldsymbol{h_\theta}(\bar{\boldsymbol{v}}) = \boldsymbol{m_\theta}(\bar{\boldsymbol{v}};\boldsymbol{\mu}).$$

The a priori learning problem has a similar form, with the important difference that the observation equation (36) changes to $\boldsymbol{y} = \boldsymbol{h_\theta}(\bar{\boldsymbol{u}}) + \varepsilon$, which can be solved *independently* from the forward model (35) (see Figure 1).

The formulation above can aid in employing concepts from the inverse problem community, like regularization, that address the ill-posedness of the learning problem [168, 169, 170]. For example, a regularized inverse problem could read

$$(39) \qquad \min_\theta \|\boldsymbol{y} - \boldsymbol{h_\theta}(\bar{\boldsymbol{v}})\|^2 + \alpha T(\boldsymbol{\theta}),$$

with a regularization function T, e.g., $T = \|\boldsymbol{\theta}\|^2$, and a hyperparameter α. In addition, the connection between machine learning in general and inverse problems has been the focus of many efforts. For instance, derivative-free optimization of neural network parameters (i.e., without back-propagation) has been achieved by means of a modified ensemble Kalman filter [171] and ensemble Kalman inversion [172]. This can potentially mitigate the need for differentiable solvers in a posterior learning. Data-driven parameterization studies (in the context of weather and climate predictions) have benefited from the ensemble Kalman inversion (EKI) approach. Lopez-Gomez et al. [173] posed the learning of SGS turbulence and convection as a regularized inverse problem and used EKI and unscented Kalman inversion to solve it. In [174], EKI is used to re-train two layers of a previously offline-trained neural network model for gravity wave forcing.

6.2. **PDE-constrained optimization.** Because the minimization formulation of the inverse problem, equation (39), is constrained to the (generally) nonlinear forward model dynamics—which is typically a PDE—it is a PDE-constrained optimization problem [84, 150]. This viewpoint, and the realization that the dimension of the parameter space is typically very large, naturally leads to the consideration of adjoint methods that compute the gradient $\frac{\partial J_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}}$ of the objective function $J_{\boldsymbol{\theta}} = \|\boldsymbol{y} - \boldsymbol{h}_{\boldsymbol{\theta}}(\bar{\boldsymbol{v}})\|^2$. Formulating the adjoint equations, discretizing them, and solving the optimization problem is the DPM approach taken in [103], for example. Alternatively, one can first discretize the PDE and use back-propagation (reverse-mode automatic differentiation) through the entire PDE solver to get gradients of the cost function with respect to the parameters [80, 77] (see Section 3.2).

6.3. **Data assimilation.** Another useful perspective is to view the closure problem from the angle of data assimilation [175, 3]. In variational data assimilation, the formulation is similar to equations (35) and (36) (but formulated for the discrete (finite-dimensional) system), with several main differences: observations are not high-fidelity simulations but actual measurements; the parameters are not neural network parameters but initial conditions; and the optimization problem is split into sequential steps, corresponding to temporal windows for which the measurement data are becoming available.

The data assimilation perspective is useful because it provides a natural approach to extend the learning of closure models by high-fidelity data to include (streaming) measurement data. The parameters of closure models can be seen as time-dependent quantities that are changing while high-fidelity or measurement data arrives, in line with the online learning approaches of [46] and [69], and the reinforcement learning approach explained in 3.5. The data assimilation perspective and PDE-constrained optimization formulation naturally connect to the concept of optimal control. In optimal control, one seeks (time-dependent) parameters that minimize an objective function, subject to given dynamics.

6.4. **Statistical approaches.** The approaches in Sections 6.1, 6.2, and 6.3 are variational in the sense that a minimization problem is solved. They provide a single-point estimate of the closure term. This is the most common approach in the closure modeling community. A logical alternative is to consider statistical approaches like Bayesian methods, which can naturally include prior knowledge of the parameters. Bayesian methods treat the parameters of interest as random variables and provide a full probabilistic description of the knowledge about the learned parameters. Such methods have the advantage that they naturally come with an expression for the uncertainty in the parameters. However, their computational expense (typically due to sampling the posterior) is high. In the data assimilation community, the use of statistical methods instead of variational methods is in fact also common [175]. Before the rise of scientific machine learning methods, Bayesian methods have been applied to learn parameters of closure models, in particular of RANS model coefficients [176, 52].

In the context of scientific machine learning, the parameters of interest in the Bayesian framework can be either the neural network weights $\boldsymbol{\theta}$, as in Bayesian neural networks [177, 178], or the model output itself (e.g., the closure term $\boldsymbol{m}_{\boldsymbol{\theta}}$). In [179], a Bayesian inference framework is developed to learn the distribution of neural network weights, with application to SGS parameterization in the Lorenz 96 model. The statistics of the predicted parameterizations can be approximated by sampling from the posterior distribution of the network parameters. A simpler way to estimate the uncertainty in the model's prediction is through deep ensemble methods [180], where an ensemble is formed by training the same model multiple times with different initializations. Such an approach was employed in [141, 181] to model the uncertainty in the learned closure model in LES and ROM contexts. Alternatively, the ensemble can be created by varying the model choices (e.g., neural network architectures), training data, the optimization algorithm, dropouts, etc. In addition, machine learning models can be trained to directly predict statistical measures of the closure terms (e.g., mean and variance). Then, the loss function may be written in terms of the log-likelihood of the corresponding distribution. The combination of data assimilation and machine learning in a Bayesian setting has been explored in the more general context of learning dynamics [182]. In the closure modeling context, this has only being explored recently [183].

7. Non-locality and the Mori-Zwanzig viewpoint

In general, because closure assumes that certain degrees of freedom will not be represented explicitly (either due to necessity or efficiency), its representation can lead to *non-local* effects for the large-scale or resolved degrees of freedom. As an example, in the case of closure for a time-dependent problem due to a coarse spatial resolution, one may need to use neighboring (or even further apart) spatio-temporal information about a resolved point when constructing its closure. We review temporal and spatial non-local closures separately, while keeping in mind that they can also appear simultaneously.

7.1. **Non-locality in time.** The Mori–Zwanzig (MZ) formalism is a general framework for treating temporal non-locality. It originated in non-equilibrium statistical mechanics as a way to construct transport equations [184] and was later reformulated as a model reduction (closure) approach [185, 186, 187] for systems of ordinary differential equations. It is based on the idea that instead of constructing a reduced model directly at the level of the ordinary differential equations (which are usually nonlinear and thus not easily amenable to projection approaches), one can construct a *linear* PDE for each resolved degree of freedom.

For the sake of concreteness, let us assume like in Section 5.1 that we have performed only spatial discretization for a time-dependent PDE and that we have obtained a system of ODEs

$$(40) \qquad \frac{\mathrm{d}\boldsymbol{u}_h}{\mathrm{d}t} = \boldsymbol{r}_h(\boldsymbol{u}_h; \boldsymbol{\mu}),$$

where $\boldsymbol{u}_h \approx \boldsymbol{u}(\boldsymbol{x}, t)$ is the vector denoting the spatially discretized solution using resolution h, resulting in M degrees of freedom. Also, let $\boldsymbol{u}_h^0 \approx \boldsymbol{u}(\boldsymbol{x}, 0)$ be the vector denoting the discretization of the initial condition. To avoid confusion for the reader, we note that due to the usual convention of presentation of the MZ formalism, the dynamics of the high-fidelity model are written as in (40) instead of the residual form $\frac{\mathrm{d}\boldsymbol{u}_h}{\mathrm{d}t} + \boldsymbol{f}_h(\boldsymbol{u}_h; \boldsymbol{\mu}) = 0$, which we have used so far in the text. Thus, $\boldsymbol{r}_h(\boldsymbol{u}_h; \boldsymbol{\mu}) = -\boldsymbol{f}_h(\boldsymbol{u}_h; \boldsymbol{\mu})$.

According to the MZ formalism, the M-dimensional system of ODEs (40) can be transformed into a system of M linear PDEs, one for each spatial-discretization point [184]. Using semigroup notation, the linear PDEs are given by

$$(41) \qquad \frac{\partial}{\partial t} e^{tL} \boldsymbol{u}_{hk}^0 = L e^{tL} \boldsymbol{u}_{hk}^0 \qquad \text{for} \qquad k = 1, \ldots, M,$$

where

$$(42) \qquad L = \sum_{j \in M} \boldsymbol{r}_{hj}(\boldsymbol{u}_h^0) \frac{\partial}{\partial \boldsymbol{u}_{hj}^0}$$

is called the Liouville operator and $e^{tL} \boldsymbol{u}_{hk}^0 = \boldsymbol{u}_{hk}(t)$.

Suppose that the vector of initial conditions can be divided as $\boldsymbol{u}_h^0 = (\bar{\boldsymbol{u}}_h^0, \tilde{\boldsymbol{u}}_h^0)$, where $\bar{\boldsymbol{u}}_h^0$ is the vector of the resolved (typically large-scale) variables and $\tilde{\boldsymbol{u}}_h^0$ is the vector of the unresolved (typically small-scale) variables. Let P be an orthogonal projection on the space of functions of $\bar{\boldsymbol{u}}_h^0$ and $Q = I - P$.

Equation (41) can be rewritten as

$$(43) \qquad \frac{\partial}{\partial t} e^{tL} \boldsymbol{u}_{hk}^0 = e^{tL} PL \boldsymbol{u}_{hk}^0 + e^{tQL} QL \boldsymbol{u}_{hk}^0 + \int_0^t e^{(t-s)L} PL e^{sQL} QL \boldsymbol{u}_{hk}^0 \mathrm{d}s,$$

where we have used Dyson's formula

$$(44) \qquad e^{tL} = e^{tQL} + \int_0^t e^{(t-s)L} PL e^{sQL} \mathrm{d}s.$$

Equation (43) is the MZ identity. It decomposes the right-hand-side of the PDE system (41) into three terms: (1) a term involving only the values of the resolved degrees of freedom at the current time (usually called the "Markovian" term), (2) a term that evolves in the space orthogonal to the resolved degrees of freedom (usually called the "noise" term), and (3) a term that accounts for the history of the interaction between resolved and unresolved degrees of freedom (usually called the "memory" term). The MZ formalism is based on an exact reformulation of the original system and though almost never used in its full generality, it is a good starting point for approximations based on mathematical, physical, and numerical considerations. Specifically, it is able to address cases

with very short, moderate, or very long memories. A detailed presentation of the MZ formalism and its relation to other formalisms developed for specific memory types can be found in [188]. Also, we must note that while we have presented the MZ formalism for the case of a system of ODEs, the formalism has also been developed for systems of stochastic differential equations—see, e.g., [189, 190, 191].

The MZ formalism, through various approximations, has been used to obtain reduced order models and closures for a host of systems (see, e.g., [192, 193, 194, 195, 196, 197, 198, 199, 200]). However, except for special cases, it is difficult to guarantee the stability of reduced order models. A renormalized version of the MZ formalism has been introduced (inspired by renormalization methods in physics), which has allowed the stabilization of such models [201, 202, 203, 204].

Finally, in recent years, data-driven approaches to MZ have also emerged. These approaches use neural networks to develop frameworks that are inspired or can be mapped to MZ [205, 206, 207, 208, 209, 210, 211, 62, 212]. Also, there exist approaches that represent one or more parts of the MZ formalism (i.e., Markovian term, memory, and/or noise). These range from the use of effective machine-learned potentials [213, 214], to neural network representation of the memory [215, 216, 217, 218, 219, 143, 220], to statistical-informed neural networks in order to simulate the noise [221]. The success of the approaches highlights the need for significant amounts of training data, the right choice of neural network architecture for the calculation of the various MZ terms, as well as the expected issue of guaranteeing stability for the effective equations for the resolved degrees of freedom.

7.2. **Non-locality in space.** Non-locality in space has been used to model phenomena that are not amenable to modeling, e.g., with integer-order derivatives. Non-locality in space aims to accommodate two types of issues: (1) rough solutions that may not be differentiable, and (2) long-range spatial correlations at the scale where we represent a phenomenon. There are two main approaches to account for non-locality in space—(1) the use of integral terms (of convolutional type), e.g., like in peridynamics, and (2) the use of fractional order derivatives (see e.g. [222, 223, 224, 225, 226].

The use of closure exhibiting non-local spatial features is a topic of active research and includes traditional (see e.g., [227, 228, 229] and machine-learning approaches. The machine-learning approaches range from techniques to estimate the various exponents appearing in the relations representing the non-local terms [230, 231], to using neural networks to represent the kernels appearing in the non-local expressions [232, 233, 234], to the use of neural networks to represent the whole non-local term [235, 71, 236, 67, 237, 238]. The approaches again highlight the need for accurate training data, the right choice of neural network architecture, and the careful coupling with the rest of the dynamics present in the model.

8. Multi-fidelity and multiscale viewpoints

8.1. **Multi-fidelity.** The process of closure modeling and surrogate modeling almost always involves collecting curated datasets of input-output responses, either through high-fidelity simulations or through experiments (one exception is PINN [24]). However, in practice, the incurred costs of data collection can restrict the quantity and quality of simulations or experiments. Building a model solely from a small high-fidelity dataset or a large low-fidelity dataset is not generally adequate. Multi-fidelity modeling approaches aim to close this gap by leveraging both low- and high-fidelity components to achieve levels of accuracy and efficiency that are not attainable otherwise. The term "component" here refers to the fact that different fidelities might appear in terms of collected data, model assumptions, physical complexity, numerical schemes, etc. For example, Howard et al. [239] showed that a multi-fidelity DeepONet can be trained by combining low-fidelity data with knowledge of physics (in terms of the governing PDE) as the high-fidelity portion.

Although the current review focuses on approaches that improve the estimation of the *reduced* quantity $\bar{\boldsymbol{u}}$ (through closure modeling), we note that the vast majority of literature on multi-fidelity modeling is instead focused on directly estimating the quantity of interest $\boldsymbol{u}$. In other words, the low-fidelity model provides $\boldsymbol{u}_{low}$ and the high-fidelity model gives $\boldsymbol{u}_{high}$, representing the low- and high-fidelity approximations of the same quantity $\boldsymbol{u}$. Then, frameworks are built such that the majority of computations are delegated to low-fidelity models whose outputs can be corrected using a small number of high-fidelity runs. Multi-fidelity modeling approaches have been particularly

beneficial for outer-loop applications (e.g., uncertainty quantification and design optimization)—for a survey of such methods, see [240]. Different studies have considered different variants for the low-fidelity model and how it is embedded in the process of learning high-fidelity outputs. For example, Conti et al. [241] developed a long short-term memory (LSTM) framework to combine abundant low-fidelity data and limited high-fidelity data, where the low-fidelity data are generated from either a pretrained ML model or using coarser spatial/temporal discretizations. The quantity of interest in the multi-fidelity setting was, however, a low-dimensional vector (e.g., the drag and lift coefficients), which was also the theme in the majority of multi-fidelity surrogate modeling literature [242, 243]. Recently, convolutional neural networks (CNNs) and graph neural networks (GNNs) have alleviated this limitation and allowed for dealing with flow field inputs/outputs efficiently [244, 245, 246].

In [247], the full field prediction problem is addressed by projecting the low-fidelity solution onto a common subspace spanned by the leading proper orthogonal decomposition (POD) modes in the form of $\boldsymbol{u}_{low}(\boldsymbol{x}, t) = \sum_{i=1}^{N} a_{low,i}(t)\boldsymbol{\phi}_i(\boldsymbol{x})$. The low-fidelity POD coefficients are then passed through LSTM to learn their high-fidelity counterparts, giving a high-fidelity approximation $\boldsymbol{u}_{high}(\boldsymbol{x}, t) = \sum_{i=1}^{N} a_{high,i}(t)\boldsymbol{\phi}_i(\boldsymbol{x})$. Because POD-based approximations of $\boldsymbol{u}$ can be erroneous (even with best-fit coefficients), Demo et al. [248] proposed a multi-fidelity modeling approach to learn the projection (truncation) error in POD reconstruction of steady-state flow problems. The leading POD coefficients are computed either using interpolation from pre-collected data or through sensor measurements. Then, a DeepONet is used to estimate the difference between the true field $\boldsymbol{u}$ and its reduced version $\bar{\boldsymbol{u}}$. Geneva and Zabaras [249] proposed a multi-fidelity generative model that produces high-fidelity realizations of turbulent flows by conditioning an invertible neural network on the corresponding low-fidelity predictions.

Particularly relevant to the scope of this review is the closure learning methodology in [91], where $\boldsymbol{G}_{\boldsymbol{\theta}}(\bar{\boldsymbol{v}}; \boldsymbol{\mu}) = \boldsymbol{F}(\bar{\boldsymbol{v}}; \boldsymbol{\mu})$ is considered as the low-fidelity model for the reduced quantity (i.e., ignoring the commutator error). Then, a multi-fidelity DeepONet is trained to learn the contribution of the closure term in a predictor-corrector fashion. The authors combined this multi-fidelity viewpoint with a posteriori learning to enhance the stability of the resulting predictions. The *model fusion* approach in [250] can also be interpreted as a multi-fidelity learning approach, where the predictions of the Galerkin-POD model are used to guide a deep neural network toward an improved estimate of the leading POD coefficients. Sen et al. [251] considered a multi-fidelity modeling approach to address the closure problem in multiscale systems with distinct meso- and macroscales. They considered a shocked particulate flow where the mesoscale effects are incorporated into the macroscale model in the form of a homogenized drag value. A low-fidelity surrogate model for the drag is first developed by the modified Bayesian Kriging method using a set of low-fidelity mesoscale simulations. A multi-fidelity learning framework is subsequently constructed to correct the low-fidelity predictions needing only a few high-fidelity simulations.

It is worth noting that using a multi-fidelity learning approach over a single fidelity one should be performed carefully. For example, [252] showed that including low-fidelity samples along with the high-fidelity samples can lead to less accurate surrogates than just using the available high-fidelity samples.

8.2. **Multiscale.** Any review of closure modeling would not be complete without referencing the classical methods for reduction of systems exhibiting time-scale separation between variables that are needed to fully describe the state of the system. A comprehensive review on various methods to perform reduction for such systems can be found in [188], while more recent accounts are in [253, 2]. The main classes of methods are invariant manifolds [254], averaging [255, 256, 257], and white noise approximation [258]. We note that these methods can also be interpreted as limiting cases of the more general MZ formalism (see, e.g., [188]).

In recent years, there have appeared data-driven variants of the approaches above for systems exhibiting time-scale separation. The main topics of these approaches are the discovery of homogenized/effective equations for slow variables [259, 260], the discovery of collective variables that promote time-scale separation [41, 261, 262, 263], as well as the estimation of closure terms to account for the effect of fast variables on slow variables [264, 265, 266, 267, 262, 268, 260].

9. Outlook

The field of turbulence closure with machine learning has advanced rapidly in recent years, but major challenges remain [269]. Here we list a few of these, including directions for future research.

9.1. Interpretability and generalizability.
The machine-learned closure models discussed in this paper are generally represented by neural networks that do not have a clear sense of interpretability. However, interpretability is a key requirement for machine-learned closure models to find their way to engineering practice. Consequently, recent research efforts [270, 271, 272, 72] are being directed on providing interpretable, closed-form expressions for the closure model, e.g., using techniques like sparse linear regression with a physics-informed library of candidate functions. Besides physical interpretability, this approach also promises to require fewer training data and have lower training costs [271].

Next to interpretability, generalizability is an important research area that is a prerequisite for trained closure models to be adopted in various situations. Training a model for certain parameters (e.g., Reynolds number), geometries, boundary conditions, and initial conditions and using it beyond this parameter set could be seen as the "holy grail" in closure modeling research. However, this is currently out of reach in its generality. Similarly, training a model with training data that is associated to a certain grid, time step, and discretization scheme and reusing it under different conditions is a difficult but central goal in closure modeling that is drawing a lot of attention. A key issue for extrapolation in time is dealing with the chaotic nature of many physical models. Learning chaotic dynamics with machine learning has been investigated for several years [273, 274, 275, 276, 41, 277], but needs to further extend into the closure modeling community.

9.2. Reduced model forms and including known physics.
It is generally acknowledged that including known physics in data-driven methods is beneficial, reducing the amount of training data required and improving generalizability. In the closure modeling context of this paper, one would similarly argue that the closure form (11) is more physics-informed than the physics-agnostic form (9) and therefore has more potential to generalize and to train. However, the evidence that supports this claim is still rather thin (some examples are given in [77]), and some studies are pointing in the reverse direction [153, 29, 127]. Restricting the reduced models to have a closure form or a particular known symmetry was *limiting* their performance in these cases. Similarly, so-called foundation models have been recently developed in the weather forecasting community that are completely agnostic of the physics (although there is an implicit dependence on the physics through the training data that is generated from physics-based solvers) [154]. These foundation models are then fine-tuned for a specific task. It is still an open question whether such foundation models could play a role in a closure model context, or whether they could even make the entire concept of closure models redundant. Perhaps a 'foundation closure model' (associated to a certain PDE) could be constructed, which is fine-tuned for a specific prediction task (boundary conditions, geometry, parameters, etc.). However, for a PDE like the Navier-Stokes equations, this seems at present an insurmountable task, given the effectively inexhaustible richness of possible flow configurations. In contrast to the 'top-down' approach of foundation models, 'bottom-up' approaches like the building-block model presented in [278] have been developed. These new approaches are still in the early phase of development, and which approach will stand the test of time will likely depend on a multitude of factors, including the availability of high-quality training data and access to computational resources.

9.3. Stability of hybrid models.
Scientific machine learning for closure leads to hybrid models that couple traditional solvers with neural networks. The often black-box nature of the neural network component can hinder the study of stability properties of the hybrid model, and, even worse, can amplify the risk of catastrophic instabilities. The problem of instabilities for coupled models has long been known (see, e.g., [279] and references therein) and is not specific to the use of neural network closures. For neural network-based closures, possible overfitting of the training data can exacerbate the instability problem [161]. Approaches like a posteriori learning presented in Section 3.2 that rely on training the neural network so that the hybrid model accurately predicts a rolled-out trajectory of certain length have shown promise. However, thorough investigation of the stability properties of such approaches is still an active research topic.

9.4. **Network architectures.** Choosing the right network architecture and optimizing its parameters is often more of an art than a science. In this paper, little attention has been paid to the different network architectures used in the context of closure modeling because new architectures are appearing continuously and because it is still highly non-trivial which one to choose. Early machine learning-based closure modeling efforts focused on simple neural network layouts (fully connected multi-layer perceptrons) [49, 59], while later convolutional neural networks and autoencoders became popular approaches. Convolutional networks are useful for structured data and mimic, to a certain extent, finite difference operators, which is why they work well in the context of differential equations. Autoencoders are appropriate in the context of compression, for example in reduced order models. More recently, graph neural nets [280, 156] have gained attention because they work on unstructured data, and neural operators because of their grid independence [63, 32, 33, 34, 35, 167]. Only a few studies investigate the effect of the network architecture and the associated optimization process, e.g., [281].

9.5. **Benchmarking, test cases, and datasets.** Currently, many papers consider simplified PDE problems as test cases, such as the Burgers' equation, the Kuramoto-Sivashinsky equation, the Lorenz '96 equations, or the incompressible Navier-Stokes equations on periodic domains with low Reynolds numbers. A number of efforts have been initiated to build structured datasets that can be used to train machine learning methods [282, 283, 284, 285, 286, 287]. Because neural network design and training often require a lot of experience and trial and error, well-defined benchmarks, documented datasets, and relevant error metrics are of utmost importance. Extension to more complex physics is still an open topic—in fluid flows, closure models for compressible flows with shock waves, for irregular domains, for reactive flows, and for high Reynolds number flows are a sparsely explored but important territory [136, 288]. Part of benchmarking also means a fair assessment on whether large neural network models (and the associated training time and data requirements) can outperform classical (PDE-based) solvers in terms of accuracy per operation count or runtime. The choice of the reference solver that we compare against and try to beat is also a critical aspect. Researchers might be tempted to compare a neural-PDE solver against a classical PDE solver that is significantly more accurate and more expensive to justify the accuracy-cost trade-off of the former. However, in many cases, one can find another low-fidelity PDE solver that is as accurate as the neural-PDE solver but less-expensive. Thus, a fair comparison should involve a Pareto front analysis that surveys all solvers that are at our disposal. In addition, many data-driven methods (for closures or otherwise) are limited by errors resulting from distribution shifting when deployed on test data that is significantly different from the training data. Learning useful 'invariant' input features may assist with this but that is as yet a largely unsolved problem in machine learning [289].

program under the DeepFusion Accelerator for Fusion Energy Sciences in Disruption Mitigations project (FOA-2905).

References

[1] H. Christensen, L. Zanna, Parametrization in Weather and Climate Models, Oxford University Press, 2022. `doi:10.1093/acrefore/9780190228620.013.826`.

[2] W. E, Principles of Multiscale Modeling, Cambridge University Press, Cambridge ; New York, 2011.

[3] S. E. Ahmed, S. Pawar, O. San, A. Rasheed, T. Iliescu, B. R. Noack, On closures for reduced order models—a spectrum of first-principle to machine-learned avenues, Physics of Fluids 33 (9) (2021).

[4] C. D. Levermore, Moment closure hierarchies for kinetic theories, Journal of Statistical Physics 83 (5) (1996) 1021–1065. `doi:10.1007/BF02179552`.

[5] J. Han, C. Ma, Z. Ma, W. E, Uniformly accurate machine learning-based hydrodynamic models for kinetic equations, Proceedings of the National Academy of Sciences 116 (44) (2019) 21983–21991. `doi:10.1073/pnas.1909854116`.

[6] J. Huang, Y. Cheng, A. J. Christlieb, L. F. Roberts, Machine learning moment closure models for the radiative transfer equation I: Directly learning a gradient based closure, Journal of Computational Physics 453 (2022) 110941. `doi:10.1016/j.jcp.2022.110941`.

[7] H. Frezat, G. Balarac, J. L. Sommer, R. Fablet, Gradient-free online learning of subgrid-scale dynamics with neural emulators (Oct. 2023). `arXiv:2310.19385`, `doi:10.48550/arXiv.2310.19385`.

[8] M. Sonnewald, R. Lguensat, D. C. Jones, P. D. Dueben, J. Brajard, V. Balaji, Bridging observations, theory and numerical simulation of the ocean using machine learning, Environmental Research Letters 16 (7) (2021) 073008. `doi:10.1088/1748-9326/ac0eb0`.

[9] T. Bolton, L. Zanna, Applications of Deep Learning to Ocean Data Inference and Subgrid Parameterization, Journal of Advances in Modeling Earth Systems 11 (1) (2019) 376–399. `doi:10.1029/2018MS001472`.

[10] M. Christie, Upscaling for Reservoir Simulation, J Pet Technol 48 (11) (1996) 1004–1010. `doi:10.2118/37324-JPT`.

[11] M. G. Saunders, G. A. Voth, Coarse-Graining Methods for Computational Biology, Annual Review of Biophysics 42 (1) (2013) 73–93. `doi:10.1146/annurev-biophys-083012-130348`.

[12] E. van der Giessen, P. A. Schultz, N. Bertin, V. V. Bulatov, W. Cai, G. Csányi, S. M. Foiles, M. G. D. Geers, C. González, M. Hütter, W. K. Kim, D. M. Kochmann, J. LLorca, A. E. Mattsson, J. Rottler, A. Shluger, R. B. Sills, I. Steinbach, A. Strachan, E. B. Tadmor, Roadmap on multiscale materials modeling, Modelling and Simulation in Materials Science and Engineering 28 (4) (2020) 043001. `doi:10.1088/1361-651X/ab7150`.

[13] K. Duraisamy, G. Iaccarino, H. Xiao, Turbulence Modeling in the Age of Data, Annual Review of Fluid Mechanics 51 (1) (2019) 357–377. `doi:10.1146/annurev-fluid-010518-040547`.

[14] S. L. Brunton, J. N. Kutz, Machine Learning for Partial Differential Equations (Mar. 2023). `arXiv:2303.17078`, `doi:10.48550/arXiv.2303.17078`.

[15] M. Lino, S. Fotiadis, A. A. Bharath, C. D. Cantwell, Current and emerging deep-learning methods for the simulation of fluid dynamics, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 479 (2275) (2023) 20230058. `doi:10.1098/rspa.2023.0058`.

[16] J. Viquerat, P. Meliga, A. Larcher, E. Hachem, A review on deep reinforcement learning for fluid mechanics: An update, Physics of Fluids 34 (11) (2022) 111301. `doi:10.1063/5.0128446`.

[17] P. Sagaut, Large Eddy Simulation for Incompressible Flows: An Introduction, 3rd Edition, Scientific Computation, Springer, Berlin ; New York, 2006.

[18] J. S. Hesthaven, G. Rozza, B. Stamm, Certified Reduced Basis Methods for Parametrized Partial Differential Equations, SpringerBriefs in Mathematics, Springer International Publishing, Cham, 2016. `doi:10.1007/978-3-319-22470-1`.

[19] S. L. Brunton, J. L. Proctor, J. N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proceedings of the national academy of sciences 113 (15) (2016) 3932–3937.

[20] S. H. Rudy, S. L. Brunton, J. L. Proctor, J. N. Kutz, Data-driven discovery of partial differential equations, Science Advances 3 (4) (2017) e1602614. `doi:10.1126/sciadv.1602614`.

[21] J.-C. Loiseau, B. R. Noack, S. L. Brunton, Sparse reduced-order modelling: sensor-based dynamics to full-state estimation, Journal of Fluid Mechanics 844 (2018) 459–490.

[22] H. Vaddireddy, A. Rasheed, A. E. Staples, O. San, Feature engineering and symbolic regression methods for detecting hidden physics from sparse sensor observation data, Physics of Fluids 32 (1) (2020).

[23] K. Fukami, T. Murata, K. Zhang, K. Fukagata, Sparse identification of nonlinear dynamics with low-dimensionalized flow representations, Journal of Fluid Mechanics 926 (2021) A10.

[24] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686–707. `doi:10.1016/j.jcp.2018.10.045`.

[25] W. E, B. Yu, The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems, Communications in Mathematics and Statistics 6 (1) (2018) 1–12. `doi:10.1007/s40304-018-0127-z`.

[26] M. Kumar, N. Yadav, Multilayer perceptrons and radial basis function neural network methods for the solution of differential equations: A survey, Computers & Mathematics with Applications 62 (10) (2011) 3796–3811. `doi:10.1016/j.camwa.2011.09.028`.

[27] S. Cai, Z. Mao, Z. Wang, M. Yin, G. E. Karniadakis, Physics-informed neural networks (PINNs) for fluid mechanics: A review, Acta Mechanica Sinica 37 (12) (2021) 1727–1738. doi:10.1007/s10409-021-01148-1.

[28] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nature Reviews Physics 3 (6) (2021) 422–440.

[29] K. Stachenfeld, D. B. Fielding, D. Kochkov, M. Cranmer, T. Pfaff, J. Godwin, C. Cui, S. Ho, P. Battaglia, A. Sanchez-Gonzalez, Learned Coarse Models for Efficient Turbulence Simulation (Apr. 2022). arXiv:2112. 15275, doi:10.48550/arXiv.2112.15275.

[30] J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Azizzadenesheli, P. Hassanzadeh, K. Kashinath, A. Anandkumar, FourCastNet: A Global Data-driven High-resolution Weather Model using Adaptive Fourier Neural Operators (Feb. 2022). arXiv:2202.11214, doi:10.48550/arXiv.2202.11214.

[31] R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, O. Vinyals, J. Stott, A. Pritzel, S. Mohamed, P. Battaglia, GraphCast: Learning skillful medium-range global weather forecasting (Aug. 2023). arXiv: 2212.12794, doi:10.48550/arXiv.2212.12794.

[32] L. Lu, P. Jin, G. Pang, Z. Zhang, G. E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nature Machine Intelligence 3 (3) (2021) 218–229. doi:10.1038/s42256-021-00302-5.

[33] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier Neural Operator for Parametric Partial Differential Equations (May 2021). arXiv:2010.08895, doi:10.48550/arXiv. 2010.08895.

[34] Q. Cao, S. Goswami, G. E. Karniadakis, Lno: Laplace neural operator for solving differential equations, arXiv preprint arXiv:2303.10528 (2023).

[35] M. Xu, J. Han, A. Lou, J. Kossaifi, A. Ramanathan, K. Azizzadenesheli, J. Leskovec, S. Ermon, A. Anandkumar, Equivariant graph neural operator for modeling 3d dynamics, arXiv preprint arXiv:2401.11037 (2024).

[36] B. Raonic, R. Molinaro, T. De Ryck, T. Rohner, F. Bartolucci, R. Alaifari, S. Mishra, E. de Bézenac, Convolutional Neural Operators for robust and accurate learning of PDEs, Advances in Neural Information Processing Systems 36 (Feb. 2024).

[37] E. Dupont, A. Doucet, Y. W. Teh, Augmented Neural ODEs, in: Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019.

[38] H. A. Melchers, Machine learning for closure models, Master's thesis, Eindhoven University of Technology (Jun. 2022).

[39] T. van Gastelen, W. Edeling, B. Sanderse, Energy-Conserving Neural Network for Turbulence Closure Modeling (Feb. 2023). arXiv:2301.13770, doi:10.48550/arXiv.2301.13770.

[40] I. G. Kevrekidis, C. W. Gear, J. M. Hyman, P. G. Kevrekidis, O. Runborg, C. Theodoropoulos, Equation-Free, Coarse-Grained Multiscale Computation: Enabling Microscopic Simulators to Perform System-Level Analysis, Communications in Mathematical Sciences 1 (4) (2003) 715–762.

[41] P. R. Vlachas, G. Arampatzis, C. Uhler, P. Koumoutsakos, Multiscale simulations of complex systems by learning their effective dynamics, Nature Machine Intelligence 4 (4) (2022) 359–366. doi:10.1038/s42256-022-00464-w.

[42] B. Tzen, M. Raginsky, Neural Stochastic Differential Equations: Deep Latent Gaussian Models in the Diffusion Limit (Oct. 2019). arXiv:1905.09883, doi:10.48550/arXiv.1905.09883.

[43] A. Boral, Z. Y. Wan, L. Zepeda-Núñez, J. Lottes, Q. Wang, Y.-f. Chen, J. R. Anderson, F. Sha, Neural Ideal Large Eddy Simulation: Modeling Turbulence with Neural Stochastic Differential Equations (Jun. 2023). arXiv:2306.01174, doi:10.48550/arXiv.2306.01174.

[44] L. C. Berselli, T. Iliescu, W. J. Layton, Mathematics of Large Eddy Simulation of Turbulent Flows, Springer, 2006.

[45] J. F. MacArt, J. Sirignano, J. B. Freund, Embedded training of neural-network sub-grid-scale turbulence models, arXiv:2105.01030 [physics] (May 2021). arXiv:2105.01030.

[46] S. Rasp, Coupled online learning as a way to tackle instabilities and biases in neural network parameterizations: General algorithms and Lorenz 96 case study (v1.0), Geoscientific Model Development 13 (5) (2020) 2185–2196. doi:10.5194/gmd-13-2185-2020.

[47] B. Peherstorfer, K. Willcox, Data-driven operator inference for nonintrusive projection-based model reduction, Computer Methods in Applied Mechanics and Engineering 306 (2016) 196–215. doi:10.1016/j.cma.2016.03. 025.

[48] B. Kramer, B. Peherstorfer, K. E. Willcox, Learning Nonlinear Reduced Models from Data with Operator Inference, Annual Review of Fluid Mechanics 56 (1) (2024) 521–548. doi:10.1146/annurev-fluid-121021-025220.

[49] M. Gamahara, Y. Hattori, Searching for turbulence models by artificial neural network, Physical Review Fluids 2 (5) (2017) 054604. doi:10.1103/PhysRevFluids.2.054604.

[50] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, Journal of Fluid Mechanics 807 (2016) 155–166. doi:10.1017/jfm.2016.615.

[51] J. Ling, R. Jones, J. Templeton, Machine learning strategies for systems with invariance properties, Journal of Computational Physics 318 (2016) 22–35. doi:10.1016/j.jcp.2016.05.003.

[52] H. Xiao, J.-L. Wu, J.-X. Wang, R. Sun, C. Roy, Quantifying and reducing model-form uncertainties in Reynolds-averaged Navier–Stokes simulations: A data-driven, physics-informed Bayesian approach, Journal of Computational Physics 324 (2016) 115–136. doi:10.1016/j.jcp.2016.07.038.

[53] J.-X. Wang, J.-L. Wu, H. Xiao, A Physics Informed Machine Learning Approach for Reconstructing Reynolds Stress Modeling Discrepancies Based on DNS Data, Physical Review Fluids 2 (3) (2017) 034603. `arXiv:1606.07987, doi:10.1103/PhysRevFluids.2.034603`.

[54] M. Schmelzer, R. Dwight, P. Cinnella, Data-Driven Deterministic Symbolic Regression of Nonlinear Stress-Strain Relation for RANS Turbulence Modelling, in: 2018 Fluid Dynamics Conference, American Institute of Aeronautics and Astronautics, Atlanta, Georgia, 2018. `doi:10.2514/6.2018-2900`.

[55] A. Beck, D. Flad, C.-D. Munz, Deep neural networks for data-driven LES closure models, Journal of Computational Physics 398 (2019) 108910. `doi:10.1016/j.jcp.2019.108910`.

[56] W. N. Edeling, Quantification of Modelling Uncertainties in Turbulent Flow Simulations, Ph.D. thesis, Delft University of Technology (2015).

[57] X. I. A. Yang, S. Zafar, J.-X. Wang, H. Xiao, Predictive large-eddy-simulation wall modeling via physics-informed neural networks, Physical Review Fluids 4 (3) (2019) 034602. `doi:10.1103/PhysRevFluids.4.034602`.

[58] J. Park, H. Choi, Toward neural-network-based large eddy simulation: Application to turbulent channel flow, Journal of Fluid Mechanics 914 (2021) A16. `doi:10.1017/jfm.2020.931`.

[59] R. Maulik, O. San, A. Rasheed, P. Vedula, Subgrid modelling for two-dimensional turbulence using neural networks, Journal of Fluid Mechanics 858 (2019) 122–144. `doi:10.1017/jfm.2018.770`.

[60] M. Kurz, A. Beck, A machine learning framework for LES closure terms, ETNA - Electronic Transactions on Numerical Analysis 56 (2022) 117–137. `doi:10.1553/etna_vol56s117`.

[61] Y. Guan, A. Subel, A. Chattopadhyay, P. Hassanzadeh, Learning physics-constrained subgrid-scale closures in the small-data regime for stable and accurate LES, Physica D: Nonlinear Phenomena 443 (2023). `doi:10.1016/j.physd.2022.133568`.

[62] W. E, H. Lei, P. Xie, L. Zhang, Machine learning-assisted multi-scale modeling, Journal of Mathematical Physics 64 (7) (2023) 071101. `doi:10.1063/5.0149861`.

[63] B. Lütjens, C. H. Crawford, C. D. Watson, C. Hill, D. Newman, Multiscale Neural Operator: Learning Fast and Grid-independent PDE Solvers (Jul. 2022). `arXiv:2207.11417, doi:10.48550/arXiv.2207.11417`.

[64] S. Rasp, M. S. Pritchard, P. Gentine, Deep learning to represent subgrid processes in climate models, Proceedings of the National Academy of Sciences 115 (39) (2018) 9684–9689. `doi:10.1073/pnas.1810286115`.

[65] K. Duraisamy, Perspectives on Machine Learning-augmented Reynolds-averaged and Large Eddy Simulation Models of Turbulence, Physical Review Fluids 6 (5) (2021) 050504. `arXiv:2009.10675, doi:10.1103/PhysRevFluids.6.050504`.

[66] M. Kurz, A. Beck, Investigating Model-Data Inconsistency in Data-Informed Turbulence Closure Terms, 14th WCCM-ECCOMAS Congress 2020 (Mar. 2021). `doi:10.23967/wccm-eccomas.2020.115`.

[67] Y. Guan, A. Chattopadhyay, A. Subel, P. Hassanzadeh, Stable a posteriori LES of 2D turbulence using convolutional neural networks: Backscattering analysis and generalization to higher Re via transfer learning, Journal of Computational Physics 458 (2022) 111090. `doi:10.1016/j.jcp.2022.111090`.

[68] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, P. W. Battaglia, Learning Mesh-Based Simulation with Graph Networks (Jun. 2021). `arXiv:2010.03409, doi:10.48550/arXiv.2010.03409`.

[69] S. Pawar, O. San, Data assimilation empowered neural network parametrizations for subgrid processes in geophysical flows, Physical Review Fluids 6 (5) (2021) 050501. `doi:10.1103/PhysRevFluids.6.050501`.

[70] J. Yuval, P. A. O'Gorman, Stable machine-learning parameterization of subgrid processes for climate modeling at a range of resolutions, Nature Communications 11 (1) (2020) 3295. `doi:10.1038/s41467-020-17142-3`.

[71] A.-T. G. Charalampopoulos, T. P. Sapsis, Machine-learning energy-preserving nonlocal closures for turbulent fluid flows and inertial tracers, Physical Review Fluids 7 (2) (2022) 024305. `doi:10.1103/PhysRevFluids.7.024305`.

[72] K. Jakhar, Y. Guan, R. Mojgani, A. Chattopadhyay, P. Hassanzadeh, L. Zanna, Learning Closed-form Equations for Subgrid-scale Closures from High-fidelity Data: Promises and Challenges (Jun. 2023). `arXiv:2306.05014, doi:10.48550/arXiv.2306.05014`.

[73] M. D. Chekroun, H. Liu, K. Srinivasan, J. C. McWilliams, The high-frequency and rare events barriers to neural closures of atmospheric dynamics, Journal of Physics: Complexity 5 (2) (2024) 025004.

[74] C. Pedersen, L. Zanna, J. Bruna, P. Perezhogin, Reliable coarse-grained turbulent simulations through combined offline learning and neural emulation (Jul. 2023). `arXiv:2307.13144, doi:10.48550/arXiv.2307.13144`.

[75] T. Kirchdoerfer, M. Ortiz, Data-driven computational mechanics, Computer Methods in Applied Mechanics and Engineering 304 (2016) 81–101.

[76] K. Karapiperis, L. Stainier, M. Ortiz, J. E. Andrade, Data-driven multiscale modeling in mechanics, Journal of the Mechanics and Physics of Solids 147 (2021) 104239.

[77] H. Melchers, D. Crommelin, B. Koren, V. Menkovski, B. Sanderse, Comparison of neural closure models for discretised PDEs, Computers & Mathematics with Applications 143 (2023) 94–107. `doi:10.1016/j.camwa.2023.04.030`.

[78] A. J. Crilly, B. Duhig, N. Bouziani, Learning closure relations using differentiable programming: An example in radiation transport, Journal of Quantitative Spectroscopy and Radiative Transfer (2024) 108941`doi:10.1016/j.jqsrt.2024.108941`.

[79] X. Chen, J. Lu, G. Tryggvason, Finding closure models to match the time evolution of coarse grained 2d turbulence flows using machine learning, Fluids 7 (5) (2022) 154.

[80] B. List, L.-W. Chen, N. Thuerey, Learned turbulence modelling with differentiable fluid solvers: Physics-based loss functions and optimisation horizons, Journal of Fluid Mechanics 949 (2022) A25. `doi:10.1017/jfm.2022.738`.

[81] R. Keisler, Forecasting global weather with graph neural networks, arXiv preprint arXiv:2202.07575 (2022).

[82] A. Beck, M. Kurz, A perspective on machine learning methods in turbulence modeling, GAMM-Mitteilungen 44 (1) (Mar. 2021). `doi:10.1002/gamm.202100002`.

[83] A. Prakash, K. E. Jansen, J. A. Evans, Invariant data-driven subgrid stress modeling in the strain-rate eigenframe for large eddy simulation, Computer Methods in Applied Mechanics and Engineering 399 (2022) 115457. `doi:10.1016/j.cma.2022.115457`.

[84] M. Hinze, R. Pinneau, M. Ulbrich, S. Ulbrich, Optimization with PDE Constraints, Vol. 23 of Mathematical Modelling: Theory and Applications, Springer Netherlands, Dordrecht, 2009. `doi:10.1007/978-1-4020-8839-1`.

[85] J. Holland, J. Baeder, K. Duraisamy, Towards Integrated Field Inversion and Machine Learning With Embedded Neural Networks for RANS Modeling, in: AIAA SciTech 2019 Forum, San Diego, California, USA, 2019. `doi:10.2514/6.2019-1884`.

[86] J. Sirignano, J. F. MacArt, J. B. Freund, DPM: A deep learning PDE augmentation method with application to large-eddy simulation, Journal of Computational Physics 423 (2020) 109811. `doi:10.1016/j.jcp.2020.109811`.

[87] V. Shankar, V. Puri, R. Balakrishnan, R. Maulik, V. Viswanathan, Differentiable physics-enabled closure modeling for Burgers' turbulence (Sep. 2022). `arXiv:2209.11614`.

[88] V. Shankar, R. Maulik, V. Viswanathan, Differentiable Turbulence (Jul. 2023). `arXiv:2307.03683, doi:10.48550/arXiv.2307.03683`.

[89] H. Frezat, J. Le Sommer, R. Fablet, G. Balarac, R. Lguensat, A Posteriori Learning for Quasi-Geostrophic Turbulence Parametrization, Journal of Advances in Modeling Earth Systems 14 (11) (2022) e2022MS003124. `doi:10.1029/2022MS003124`.

[90] P. Lippe, B. S. Veeling, P. Perdikaris, R. E. Turner, J. Brandstetter, PDE-Refiner: Achieving Accurate Long Rollouts with Neural PDE Solvers (Oct. 2023). `arXiv:2308.05732, doi:10.48550/arXiv.2308.05732`.

[91] S. E. Ahmed, P. Stinis, A multifidelity deep operator network approach to closure for multiscale systems, Computer Methods in Applied Mechanics and Engineering 414 (2023) 116161. `doi:10.1016/j.cma.2023.116161`.

[92] F. Gonzalez, F.-X. Demoulin, S. Bernard, Towards Long-Term predictions of Turbulence using Neural Operators (Jul. 2023). `arXiv:2307.13517, doi:10.48550/arXiv.2307.13517`.

[93] D. Hartmann, L. Failer, A Differentiable Solver Approach to Operator Inference (Jul. 2021). `arXiv:2107.02093, doi:10.48550/arXiv.2107.02093`.

[94] W. I. T. Uy, D. Hartmann, B. Peherstorfer, Operator inference with roll outs for learning reduced models from scarce and low-quality data, Computers & Mathematics with Applications 145 (2023) 224–239. `doi:10.1016/j.camwa.2023.06.012`.

[95] S. Stolz, N. A. Adams, An approximate deconvolution procedure for large-eddy simulation, Physics of Fluids 11 (7) (1999) 1699–1701. `doi:10.1063/1.869867`.

[96] S. Pawar, O. San, A. Rasheed, P. Vedula, A priori analysis on deep learning of subgrid-scale parameterizations for Kraichnan turbulence, Theoretical and Computational Fluid Dynamics 34 (4) (2020) 429–455. `doi:10.1007/s00162-019-00512-z`.

[97] R. Maulik, O. San, A neural network approach for the blind deconvolution of turbulent flows, Journal of Fluid Mechanics 831 (2017) 151–181. `doi:10.1017/jfm.2017.637`.

[98] R. Maulik, O. San, A dynamic closure modeling framework for large eddy simulation using approximate deconvolution: Burgers equation, Cogent Physics 5 (1) (2018) 1464368. `doi:10.1080/23311940.2018.1464368`.

[99] Z. Yuan, C. Xie, J. Wang, Deconvolutional artificial neural network models for large eddy simulation of turbulence, Physics of Fluids 32 (11) (2020) 115106. `doi:10.1063/5.0027146`.

[100] Z. Yuan, Y. Wang, C. Xie, J. Wang, Dynamic iterative approximate deconvolution models for large-eddy simulation of turbulence, Physics of Fluids 33 (8) (2021) 085125. `doi:10.1063/5.0059643`.

[101] K. Fukami, K. Fukagata, K. Taira, Machine-learning-based spatio-temporal super resolution reconstruction of turbulent flows, Journal of Fluid Mechanics 909 (2021) A9. `doi:10.1017/jfm.2020.948`.

[102] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, Adaptive Computation and Machine Learning, The MIT Press, Cambridge, Massachusetts, 2016.

[103] J. Sirignano, J. F. MacArt, Dynamic Deep Learning LES Closures: Online Optimization With Embedded DNS (Mar. 2023). `arXiv:2303.02338`.

[104] W. E, J. Han, L. Zhang, Integrating Machine Learning with Physics-Based Modeling (Jun. 2020). `arXiv:2006.02619, doi:10.48550/arXiv.2006.02619`.

[105] U. Fasel, J. N. Kutz, B. W. Brunton, S. L. Brunton, Ensemble-SINDy: Robust sparse model discovery in the low-data, high-noise limit, with active learning and control, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 478 (2260) (2022) 20210904. `doi:10.1098/rspa.2021.0904`.

[106] S. Maddu, S. Weady, M. J. Shelley, Learning fast, accurate, and stable closures of a kinetic theory of an active fluid (Aug. 2023). `arXiv:2308.06675, doi:10.48550/arXiv.2308.06675`.

[107] L. Zhao, Z. Li, B. Caswell, J. Ouyang, G. E. Karniadakis, Active learning of constitutive relation from mesoscopic dynamics for macroscopic modeling of non-Newtonian flows, Journal of Computational Physics 363 (2018) 116–127. `doi:10.1016/j.jcp.2018.02.039`.

[108] G. Novati, H. L. de Laroussilhe, P. Koumoutsakos, Automating turbulence modelling by multi-agent reinforcement learning, Nature Machine Intelligence 3 (1) (2021) 87–96.

[109] H. J. Bae, P. Koumoutsakos, Scientific multi-agent reinforcement learning for wall-models of turbulent flows, Nature Communications 13 (1) (2022) 1443. `doi:10.1038/s41467-022-28957-7`.

[110] J. Kim, H. Kim, J. Kim, C. Lee, Deep reinforcement learning for large-eddy simulation modeling in wall-bounded turbulence, Physics of Fluids 34 (10) (2022) 105132. `doi:10.1063/5.0106940`.

[111] A. Beck, M. Kurz, Toward discretization-consistent closure schemes for large eddy simulation using reinforcement learning, Physics of Fluids 35 (12) (2023).

[112] P. Garnier, J. Viquerat, J. Rabault, A. Larcher, A. Kuhnle, E. Hachem, A review on Deep Reinforcement Learning for Fluid Mechanics (Feb. 2021). `arXiv:1908.04127`, `doi:10.48550/arXiv.1908.04127`.

[113] C. Meng, S. Seo, D. Cao, S. Griesemer, Y. Liu, When physics meets machine learning: A survey of physics-informed machine learning, arXiv preprint arXiv:2203.16797 (2022).

[114] V. Shankar, S. Barwey, Z. Kolter, R. Maulik, V. Viswanathan, Importance of equivariant and invariant symmetries for fluid flow modeling, arXiv preprint arXiv:2307.05486 (2023).

[115] I. E. Lagaris, A. Likas, D. I. Fotiadis, Artificial neural networks for solving ordinary and partial differential equations, IEEE transactions on neural networks 9 (5) (1998) 987–1000.

[116] H. Eivazi, M. Tahani, P. Schlatter, R. Vinuesa, Physics-informed neural networks for solving reynolds-averaged navier–stokes equations, Physics of Fluids 34 (7) (2022).

[117] S. Hanrahan, M. Kozul, R. Sandberg, Studying turbulent flows with physics-informed neural networks and sparse data, International Journal of Heat and Fluid Flow 104 (2023) 109232.

[118] V. Kag, K. Seshasayanan, V. Gopinath, Physics-informed data based neural networks for two-dimensional turbulence, Physics of Fluids 34 (5) (2022).

[119] N. Rahaman, A. Baratin, D. Arpit, F. Draxler, M. Lin, F. Hamprecht, Y. Bengio, A. Courville, On the spectral bias of neural networks, in: International Conference on Machine Learning, PMLR, 2019, pp. 5301–5310.

[120] J. Zhao, Q. Li, Mitigating distribution shift in machine learning-augmented hybrid simulation, arXiv preprint arXiv:2401.09259 (2024).

[121] J. Brandstetter, M. Welling, D. E. Worrall, Lie point symmetry data augmentation for neural pde solvers, in: International Conference on Machine Learning, PMLR, 2022, pp. 2241–2256.

[122] J. Kim, C. Lee, Deep unsupervised learning of turbulence for inflow generation at various reynolds numbers, Journal of Computational Physics 406 (2020) 109216.

[123] A. Subel, A. Chattopadhyay, Y. Guan, P. Hassanzadeh, Data-driven subgrid-scale modeling of forced burgers turbulence using deep learning with generalization to higher reynolds numbers via transfer learning, Physics of Fluids 33 (3) (2021).

[124] C. G. Speziale, Galilean invariance of subgrid-scale stress models in the large-eddy simulation of turbulence, Journal of Fluid Mechanics 156 (-1) (1985) 55. `doi:10.1017/S0022112085001987`.

[125] M. H. Silvis, R. A. Remmerswaal, R. Verstappen, Physical consistency of subgrid-scale models for large-eddy simulation of incompressible turbulent flows, Physics of Fluids 29 (1) (2017) 015105. `doi:10.1063/1.4974093`.

[126] J.-X. Wang, J. Wu, J. Ling, G. Iaccarino, H. Xiao, A comprehensive physics-informed machine learning framework for predictive turbulence modeling, arXiv preprint arXiv:1701.07102 (2017).

[127] R. Bose, A. M. Roy, Invariance embedded physics-infused deep neural network-based sub-grid scale models for turbulent flows, Engineering Applications of Artificial Intelligence 128 (2024) 107483. `doi:10.1016/j.engappai.2023.107483`.

[128] S. B. Pope, A more general effective-viscosity hypothesis, Journal of Fluid Mechanics 72 (2) (1975) 331–340.

[129] D. Razafindralandy, A. Hamdouni, Consequences of Symmetries on the Analysis and Construction of Turbulence Models, Symmetry, Integrability and Geometry: Methods and Applications (May 2006). `arXiv:physics/0605106`, `doi:10.3842/SIGMA.2006.052`.

[130] C. G. Speziale, Turbulence Modeling in Noninertial Frames of Reference, Theoretical and Computational Fluid Dynamics 1 (1) (1989) 3–19. `doi:10.1007/BF00271419`.

[131] T. Beucler, S. Rasp, M. Pritchard, P. Gentine, Achieving Conservation of Energy in Neural Network Emulators for Climate Modeling (Jun. 2019). `arXiv:1906.06622`, `doi:10.48550/arXiv.1906.06622`.

[132] K. Kashinath, P. Marcus, et al., Enforcing physical constraints in cnns through differentiable pde layer, in: ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations, 2020.

[133] M. Mohebujjaman, L. Rebholz, T. Iliescu, Physically constrained data-driven correction for reduced-order modeling of fluid flows, International Journal for Numerical Methods in Fluids 89 (3) (2019) 103–122. `doi:10.1002/fld.4684`.

[134] J. L. Callaham, S. L. Brunton, J.-C. Loiseau, On the role of nonlinear correlations in reduced-order modelling, Journal of Fluid Mechanics 938 (2022) A1. `doi:10.1017/jfm.2021.994`.

[135] E. J. Parish, K. Duraisamy, A paradigm for data-driven predictive modeling using field inversion and machine learning, Journal of computational physics 305 (2016) 758–774.

[136] V. Shankar, R. Maulik, V. Viswanathan, Differentiable turbulence ii, arXiv preprint arXiv:2307.13533 (2023).

[137] M. Kurz, P. Offenhäuser, A. Beck, Deep reinforcement learning for turbulence modeling in large eddy simulations, International Journal of Heat and Fluid Flow 99 (2023) 109094.

[138] M. Fiore, L. Koloszar, C. Fare, M. A. Mendez, M. Duponcheel, Y. Bartosiewicz, Physics-constrained machine learning for thermal turbulence modelling at low prandtl numbers, International Journal of Heat and Mass Transfer 194 (2022) 122998.

[139] R. Wang, R. Walters, R. Yu, Incorporating Symmetry into Deep Dynamics Models for Improved Generalization (Mar. 2021). `arXiv:2002.03061`, `doi:10.48550/arXiv.2002.03061`.

[140] B. Siddani, S. Balachandar, R. Fang, Rotational and reflectional equivariant convolutional neural network for data-limited applications: Multiphase flow demonstration, Physics of Fluids 33 (10) (2021) 103323. `doi:10.1063/5.0066049`.

[141] S. Pawar, O. San, A. Rasheed, P. Vedula, Frame invariant neural network closures for Kraichnan turbulence, Physica A: Statistical Mechanics and its Applications 609 (2023) 128327. `doi:10.1016/j.physa.2022.128327`.

[142] S. Kang, E. M. Constantinescu, Learning Subgrid-Scale Models in Discontinuous Galerkin Methods with Neural Ordinary Differential Equations for Compressible Navier–Stokes Equations (Oct. 2023). `arXiv:2310.18897`, `doi:10.48550/arXiv.2310.18897`.

[143] A. Gupta, P. F. J. Lermusiaux, Neural closure models for dynamical systems, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 477 (2252) (2021) 20201004. `doi:10.1098/rspa.2020.1004`.

[144] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud, Neural Ordinary Differential Equations, arXiv:1806.07366 (Dec. 2019). `arXiv:1806.07366`.

[145] P. Kidger, On Neural Differential Equations (Feb. 2022). `arXiv:2202.02435`, `doi:10.48550/arXiv.2202.02435`.

[146] C. Rackauckas, Y. Ma, J. Martensen, C. Warner, K. Zubov, R. Supekar, D. Skinner, A. Ramadhan, A. Edelman, Universal Differential Equations for Scientific Machine Learning (Nov. 2021). `arXiv:2001.04385`, `doi:10.48550/arXiv.2001.04385`.

[147] L. Pontryagin, Mathematical Theory of Optimal Processes, 1st Edition, Routledge, 1986. `doi:10.1201/9780203749319`.

[148] D. E. Kirk, Optimal Control Theory: An Introduction, Dover Publications, Mineola, N.Y, 2004.

[149] S. Massaroli, M. Poli, J. Park, A. Yamashita, H. Asama, Dissecting Neural ODEs, in: Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 3952–3963.

[150] M. D. Gunzburger, Perspectives in Flow Control and Optimization, Advances in Design and Control, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2003.

[151] D. Onken, L. Ruthotto, Discretize-Optimize vs. Optimize-Discretize for Time-Series Regression and Continuous Normalizing Flows (Jul. 2020). `arXiv:2005.13420`, `doi:10.48550/arXiv.2005.13420`.

[152] P. R. Vlachas, J. Pathak, B. R. Hunt, T. P. Sapsis, M. Girvan, E. Ott, P. Koumoutsakos, Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics, Neural Networks 126 (2020) 191–217. `doi:10.1016/j.neunet.2020.02.016`.

[153] D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, S. Hoyer, Machine learning–accelerated computational fluid dynamics, Proceedings of the National Academy of Sciences 118 (21) (2021) e2101784118. `doi:10.1073/pnas.2101784118`.

[154] C. Bodnar, W. P. Bruinsma, A. Lucic, M. Stanley, J. Brandstetter, P. Garvan, M. Riechert, J. Weyn, H. Dong, A. Vaughan, J. K. Gupta, K. Tambiratnam, A. Archibald, E. Heider, M. Welling, R. E. Turner, P. Perdikaris, Aurora: A Foundation Model of the Atmosphere (May 2024). `arXiv:2405.13063`, `doi:10.48550/arXiv.2405.13063`.

[155] Z. Li, W. Peng, Z. Yuan, J. Wang, Fourier neural operator approach to large eddy simulation of three-dimensional turbulence, Theoretical and Applied Mechanics Letters 12 (6) (2022) 100389. `doi:10.1016/j.taml.2022.100389`.

[156] J. Brandstetter, D. Worrall, M. Welling, Message Passing Neural PDE Solvers (Mar. 2023). `arXiv:2202.03376`, `doi:10.48550/arXiv.2202.03376`.

[157] J. Pathak, B. Hunt, M. Girvan, Z. Lu, E. Ott, Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach, Physical Review Letters 120 (2) (2018) 024102. `doi:10.1103/PhysRevLett.120.024102`.

[158] F. Takens, Detecting strange attractors in turbulence, in: D. Rand, L.-S. Young (Eds.), Dynamical Systems and Turbulence, Warwick 1980, Vol. 898, Springer Berlin Heidelberg, Berlin, Heidelberg, 1981, pp. 366–381. `doi:10.1007/BFb0091924`.

[159] Y. Bar-Sinai, S. Hoyer, J. Hickey, M. P. Brenner, Learning data-driven discretizations for partial differential equations, Proceedings of the National Academy of Sciences 116 (31) (2019) 15344–15349. `doi:10.1073/pnas.1814058116`.

[160] F. F. Grinstein, L. G. Margolin, W. Rider (Eds.), Implicit Large Eddy Simulation: Computing Turbulent Fluid Dynamics, first publ Edition, Cambridge Univ. Press, Cambridge, 2007.

[161] K. Um, R. Brand, Y. R. Fei, P. Holl, N. Thuerey, Solver-in-the-Loop: Learning from Differentiable Physics to Interact with Iterative PDE-Solvers, in: Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 6111–6122.

[162] G. Dresdner, D. Kochkov, P. Norgaard, L. Zepeda-Núñez, J. A. Smith, M. P. Brenner, S. Hoyer, Learning to correct spectral methods for simulating turbulent flows (Jun. 2023). `arXiv:2207.00556`, `doi:10.48550/arXiv.2207.00556`.

[163] T. Lund, The use of explicit filters in large eddy simulation, Computers & Mathematics with Applications 46 (4) (2003) 603–616. `doi:10.1016/S0898-1221(03)90019-8`.

[164] J. Gullbrand, F. K. Chow, The effect of numerical errors and turbulence models in large-eddy simulations of channel flow, with and without explicit filtering, Journal of Fluid Mechanics 495 (2003) 323–341. `doi:10.1017/S0022112003006268`.

[165] C. J. Wang, P. Golland, Discretization Invariant Learning on Neural Fields (Nov. 2022). `arXiv:2206.01178`, `doi:10.48550/arXiv.2206.01178`.

[166] N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural Operator: Learning Maps Between Function Spaces With Applications to PDEs, Journal of Machine Learning Research 24 (89) (2023) 1–97.

[167] F. Bartolucci, E. de Bezenac, B. Raonic, R. Molinaro, S. Mishra, R. Alaifari, Representation Equivalent Neural Operators: A Framework for Alias-free Operator Learning, in: Thirty-Seventh Conference on Neural Information Processing Systems, 2023.

[168] P. C. Hansen, Discrete Inverse Problems: Insight and Algorithms, Fundamentals of Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, 2010.

[169] A. M. Stuart, Inverse problems: A Bayesian perspective, Acta Numerica 19 (2010) 451–559. `doi:10.1017/S0962492910000061`.

[170] D. Calvetti, E. Somersalo, Inverse problems: From regularization to Bayesian inference, WIREs Computational Statistics 10 (3) (May 2018). `doi:10.1002/wics.1427`.

[171] E. Haber, F. Lucka, L. Ruthotto, Never look back-a modified EnKF method and its application to the training of neural networks without back propagation, arXiv preprint arXiv:1805.08034 (2018).

[172] N. B. Kovachki, A. M. Stuart, Ensemble Kalman inversion: a derivative-free technique for machine learning tasks, Inverse Problems 35 (9) (2019) 095005.

[173] I. Lopez-Gomez, C. Christopoulos, H. L. Langeland Ervik, O. R. Dunbar, Y. Cohen, T. Schneider, Training physics-based machine-learning parameterizations with gradient-free ensemble kalman methods, Journal of Advances in Modeling Earth Systems 14 (8) (2022) e2022MS003105.

[174] H. A. Pahlavan, P. Hassanzadeh, M. J. Alexander, Explainable offline-online training of neural networks for parameterizations: A 1D gravity wave-QBO testbed in the small-data regime, Geophysical Research Letters 51 (2) (2024) e2023GL106324.

[175] M. Asch, M. Bocquet, M. Nodet, Data Assimilation: Methods, Algorithms, and Applications, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2016. `doi:10.1137/1.9781611974546`.

[176] W. N. Edeling, P. Cinnella, R. P. Dwight, H. Bijl, Bayesian estimates of parameter variability in the k–ε turbulence model, Journal of Computational Physics 258 (2014) 73–94. `doi:10.1016/j.jcp.2013.10.027`.

[177] E. Goan, C. Fookes, Bayesian neural networks: An introduction and survey, Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018 (2020) 45–87.

[178] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, M. Bennamoun, Hands-on Bayesian neural networks—a tutorial for deep learning users, IEEE Computational Intelligence Magazine 17 (2) (2022) 29–48.

[179] M. A. Bhouri, P. Gentine, History-based, bayesian, closure for stochastic parameterization: Application to lorenz'96, arXiv preprint arXiv:2210.14488 (2022).

[180] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, Advances in neural information processing systems 30 (2017).

[181] S. E. Ahmed, O. San, A. Rasheed, T. Iliescu, A. Veneziani, Physics guided machine learning for variational multiscale reduced order modeling, SIAM Journal on Scientific Computing 45 (3) (2023) B283–B313.

[182] M. Bocquet, J. Brajard, A. Carrassi, L. Bertino, Bayesian inference of chaotic dynamics by merging data assimilation, machine learning and expectation-maximization, Foundations of Data Science 2 (1) (Mon Mar 23 20:00:00 EDT 2020) 55–80. `doi:10.3934/fods.2020004`.

[183] D. Maruyama, P. Bekemeyer, S. Görtz, S. Coggon, S. Sharma, Data-driven Bayesian inference of turbulence model closure coefficients incorporating epistemic uncertainty, Acta Mechanica Sinica 37 (12) (2021) 1812–1838. `doi:10.1007/s10409-021-01152-5`.

[184] R. Zwanzig, Nonequilibrium statistical mechanics, Oxford University Press, 2001.

[185] A. J. Chorin, O. H. Hald, R. Kupferman, Optimal prediction and the Mori-Zwanzig representation of irreversible processes, Proceedings of the National Academy of Sciences 97 (7) (2000) 2968–2973.

[186] A. J. Chorin, O. H. Hald, R. Kupferman, Optimal prediction with memory, Physica D: Nonlinear Phenomena 166 (3) (2002) 239–257.

[187] A. Chorin, P. Stinis, Problem reduction, renormalization, and memory, Communications in Applied Mathematics and Computational Science 1 (1) (2007) 1–27. `doi:10.2140/camcos.2006.1.1`.

[188] D. Givon, R. Kupferman, A. Stuart, Extracting macroscopic dynamics: Model problems and algorithms, Nonlinearity 17 (6) (2004) R55.

[189] L. Ma, X. Li, C. Liu, Coarse-graining langevin dynamics using reduced-order techniques, Journal of Computational Physics 380 (2019) 170–190.

[190] T. Hudson, X. H. Li, Coarse-graining of overdamped langevin dynamics via the Mori–Zwanzig formalism, Multiscale Modeling & Simulation 18 (2) (2020) 1113–1135.

[191] Y. Zhu, D. Venturi, Hypoellipticity and the Mori–Zwanzig formulation of stochastic differential equations, Journal of Mathematical Physics 62 (10) (2021).

[192] D. Bernstein, Optimal prediction of Burgers's equation, Multiscale Modeling & Simulation 6 (1) (2007) 27–52.

[193] P. Stinis, Numerical computation of solutions of the critical nonlinear Schrödinger equation after the singularity, Multiscale Modeling & Simulation 10 (1) (2012) 48–60.

[194] Z. Li, X. Bian, X. Li, G. E. Karniadakis, Incorporation of memory effects in coarse-grained modeling via the Mori-Zwanzig formalism, The Journal of Chemical Physics 143 (24) (2015) 3128.

[195] H. Lei, N. Baker, X. Li, Data-driven parameterization of the generalized Langevin equation, Proceedings of the National Academy of Sciences 33 (3) (2016) 14183–14188.

[196] E. J. Parish, K. Duraisamy, Non-Markovian Closure Models for Large Eddy Simulations using the Mori-Zwanzig Formalism, Physical Review Fluids 2 (1) (2017) 014604. `arXiv:1611.03311`, `doi:10.1103/PhysRevFluids.2.014604`.

[197] E. J. Parish, K. Duraisamy, A dynamic subgrid scale model for Large Eddy Simulations based on the Mori–Zwanzig formalism, Journal of Computational Physics 349 (2017) 154–175. `doi:10.1016/j.jcp.2017.07.053`.

[198] G. Jung, M. Hanke, F. Schmid, Iterative reconstruction of memory kernels, Journal of chemical theory and computation 13 (6) (2017) 2481–2488.

[199] S. Pan, K. Duraisamy, Data-driven Discovery of Closure Models, SIAM Journal on Applied Dynamical Systems 17 (4) (2018) 2381–2413. `arXiv:1803.09318`, `doi:10.1137/18M1177263`.

[200] B. Leimkuhler, M. Sachs, Efficient numerical algorithms for the generalized langevin equation, SIAM Journal on Scientific Computing 44 (1) (2022) A364–A388.

[201] P. Stinis, A phase transition approach to detecting singularities of partial differential equations, Communications in Applied Mathematics and Computational Science 4 (1) (2009) 217–239.

[202] P. Stinis, Renormalized Mori-Zwanzig-reduced models for systems without scale separation, in: Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, Vol. 471, 2015, p. 20140446.

[203] J. Price, P. Stinis, Renormalized reduced order models with memory for long time prediction, Multiscale Modeling & Simulation 17 (1) (2019) 68–91.

[204] J. Price, B. Meuris, M. Shapiro, P. Stinis, Optimal renormalization of multiscale systems, Proceedings of the National Academy of Sciences 118 (37) (2021) e2102266118.

[205] J. Berner, U. Achatz, L. Batté, L. Bengtsson, A. de la Cámara, H. M. Christensen, M. Colangeli, D. R. B. Coleman, D. Crommelin, S. I. Dolaptchiev, C. L. E. Franzke, P. Friederichs, P. Imkeller, H. Järvinen, S. Juricke, V. Kitsios, F. Lott, V. Lucarini, S. Mahajan, T. N. Palmer, C. Penland, M. Sakradzija, J.-S. von Storch, A. Weisheimer, M. Weniger, P. D. Williams, J.-I. Yano, Stochastic Parameterization: Toward a New View of Weather and Climate Models, Bulletin of the American Meteorological Society 98 (3) (2017) 565–588. doi:10.1175/BAMS-D-15-00268.1.

[206] C. Brennan, D. Venturi, Data-driven closures for stochastic dynamical systems, Journal of Computational Physics 372 (2018) 281–298. doi:10.1016/j.jcp.2018.06.038.

[207] S. Wiewel, M. Becher, N. Thuerey, Latent space physics: Towards learning the temporal evolution of fluid flow, in: Computer graphics forum, Vol. 38, Wiley Online Library, 2019, pp. 71–82.

[208] R. Wang, K. Kashinath, M. Mustafa, A. Albert, R. Yu, Towards Physics-informed Deep Learning for Turbulent Flow Prediction, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '20, Association for Computing Machinery, New York, NY, USA, 2020, pp. 1457–1466. doi:10.1145/3394486.3403198.

[209] K. K. Lin, F. Lu, Data-driven model reduction, Wiener projections, and the Koopman-Mori-Zwanzig formalism, Journal of Computational Physics 424 (2021) 109864.

[210] M. Levine, A. Stuart, A framework for machine learning of model error in dynamical systems, Communications of the American Mathematical Society 2 (07) (2022) 283–344.

[211] Q. Wang, N. Ripamonti, J. S. Hesthaven, Recurrent neural network closure of parametric pOD-Galerkin reduced-order models based on the Mori-Zwanzig formalism, Journal of Computational Physics 410 (2020) 109402.

[212] F. Dietrich, A. Makeev, G. Kevrekidis, N. Evangelou, T. Bertalan, S. Reich, I. G. Kevrekidis, Learning effective stochastic differential equations from microscopic simulations: Linking stochastic numerics to deep learning, Chaos: An Interdisciplinary Journal of Nonlinear Science 33 (2) (2023).

[213] J. Wang, S. Olsson, C. Wehmeyer, A. Pérez, N. E. Charron, G. De Fabritiis, F. Noé, C. Clementi, Machine learning of coarse-grained molecular dynamics force fields, ACS central science 5 (5) (2019) 755–767.

[214] A. E. Durumeric, N. E. Charron, C. Templeton, F. Musil, K. Bonneau, A. S. Pasos-Trejo, Y. Chen, A. Kelkar, F. Noé, C. Clementi, Machine learned coarse-grained protein force-fields: Are we there yet?, Current Opinion in Structural Biology 79 (2023) 102533.

[215] C. Ma, J. Wang, et al., Model reduction with memory and the machine learning of dynamical systems, arXiv preprint arXiv:1808.04258 (2018).

[216] D. Qi, J. Harlim, Machine learning-based statistical closure models for turbulent dynamical systems, Philosophical Transactions of the Royal Society A 380 (2229) (2022) 20210205.

[217] A. Russo, M. A. Durán-Olivencia, I. G. Kevrekidis, S. Kalliadasis, Machine learning memory kernels as closure for non-markovian stochastic processes, IEEE Transactions on Neural Networks and Learning Systems (2022).

[218] M. A. Bhouri, P. Gentine, Memory-based parameterization with differentiable solver: Application to Lorenz '96, Chaos: An Interdisciplinary Journal of Nonlinear Science 33 (7) (2023) 073116. doi:10.1063/5.0131929.

[219] Z. She, P. Ge, H. Lei, Data-driven construction of stochastic reduced dynamics encoded with non-markovian features, The Journal of Chemical Physics 158 (3) (2023).

[220] D. Crommelin, W. Edeling, Resampling with neural networks for stochastic parameterization in multiscale systems, Physica D: Nonlinear Phenomena 422 (2021) 132894. doi:10.1016/j.physd.2021.132894.

[221] Y. Zhu, Y.-H. Tang, C. Kim, Learning stochastic dynamics with statistics-informed neural network, Journal of Computational Physics 474 (2023) 111819.

[222] A. Javili, R. Morasata, E. Oterkus, S. Oterkus, Peridynamics review, Mathematics and Mechanics of Solids 24 (11) (2019) 3714–3739.

[223] E. Kharazmi, Z. Mao, G. Pang, M. Zayernouri, G. E. Karniadakis, Fractional calculus and numerical methods for fractional pdes, in: First Congress of Greek Mathematicians: Proceedings of the Congress Held in Athens, Walter de Gruyter GmbH & Co KG, 2020, p. 91.

[224] M. D'Elia, Q. Du, C. Glusa, M. Gunzburger, X. Tian, Z. Zhou, Numerical methods for nonlocal and fractional models, Acta Numerica 29 (2020) 1–124.

[225] J. L. Suzuki, M. Gulian, M. Zayernouri, M. D'Elia, Fractional modeling in action: A survey of nonlocal models for subsurface transport, turbulent flows, and anomalous materials, Journal of Peridynamics and Nonlocal modeling 5 (3) (2023) 392–459.

[226] Q. Du, Nonlocal modeling, analysis and computation, in: CBMS-NSF regional conference series in applied mathematics, Vol. 94, 2020.

[227] E. D. Held, J. D. Callen, C. C. Hegna, C. R. Sovinec, T. A. Gianakon, S. E. Kruger, Nonlocal closures for plasma fluid simulations, Physics of plasmas 11 (5) (2004) 2419–2426.

[228] H. Zhang, P. Qiao, Virtual crack closure technique in peridynamic theory, Computer Methods in Applied Mechanics and Engineering 372 (2020) 113318.

[229] J. Liu, H. H. Williams, A. Mani, Systematic approach for modeling a nonlocal eddy diffusivity, Physical Review Fluids 8 (12) (2023) 124501.

[230] F. Song, G. E. Karniadakis, Variable-order fractional models for wall-bounded turbulent flows, Entropy 23 (6) (2021) 782.

[231] S. H. Seyedi, M. Zayernouri, A data-driven dynamic nonlocal subgrid-scale model for turbulent flows, Physics of Fluids 34 (3) (2022).

[232] A. S. Joglekar, A. G. Thomas, Machine learning of hidden variables in multiscale fluid simulation, arXiv preprint arXiv:2306.10709 (2023).

[233] E. A. B. de Moraes, M. D'Elia, M. Zayernouri, Machine learning of nonlocal micro-structural defect evolutions in crystalline materials, Computer Methods in Applied Mechanics and Engineering 403 (2023) 115743.

[234] H. You, Y. Yu, N. Trask, M. Gulian, M. D'Elia, Data-driven learning of nonlocal physics from high-fidelity synthetic data, Computer Methods in Applied Mechanics and Engineering 374 (2021) 113553.

[235] X.-H. Zhou, J. Han, H. Xiao, Learning nonlocal constitutive models with neural networks, Computer Methods in Applied Mechanics and Engineering 384 (2021) 113927.

[236] B. Liu, H. Yu, H. Huang, N. Liu, X. Lu, Investigation of nonlocal data-driven methods for subgrid-scale stress modeling in large eddy simulation, AIP Advances 12 (6) (2022).

[237] G. Tabe Jamaat, Y. Hattori, A priori assessment of nonlocal data-driven wall modeling in large eddy simulation, Physics of Fluids 35 (5) (2023).

[238] J. Han, X.-H. Zhou, H. Xiao, An equivariant neural operator for developing nonlocal tensorial constitutive models, Journal of Computational Physics 488 (2023) 112243.

[239] A. A. Howard, M. Perego, G. E. Karniadakis, P. Stinis, Multifidelity Deep Operator Networks For Data-Driven and Physics-Informed Problems, Journal of Computational Physics 493 (2023) 112462. `arXiv:2204.09157`, `doi:10.1016/j.jcp.2023.112462`.

[240] B. Peherstorfer, K. Willcox, M. Gunzburger, Survey of multifidelity methods in uncertainty propagation, inference, and optimization, SIAM Review 60 (3) (2018) 550–591.

[241] P. Conti, M. Guo, A. Manzoni, J. S. Hesthaven, Multi-fidelity surrogate modeling using long short-term memory networks, Computer Methods in Applied Mechanics and Engineering 404 (2023) 115811.

[242] X. Zhang, F. Xie, T. Ji, Z. Zhu, Y. Zheng, Multi-fidelity deep neural network surrogate model for aerodynamic shape optimization, Computer Methods in Applied Mechanics and Engineering 373 (2021) 113485.

[243] G. Lamberti, C. Gorlé, A multi-fidelity machine learning framework to predict wind loads on buildings, Journal of Wind Engineering and Industrial Aerodynamics 214 (2021) 104647.

[244] P. Liao, W. Song, P. Du, H. Zhao, Multi-fidelity convolutional neural network surrogate model for aerodynamic optimization based on transfer learning, Physics of Fluids 33 (12) (2021).

[245] S. Mondal, S. Sarkar, Multi-fidelity prediction of spatiotemporal fluid flow, Physics of Fluids 34 (8) (2022).

[246] J. Li, Y. Li, T. Liu, D. Zhang, Y. Xie, Multi-fidelity graph neural network for flow field data fusion of turbomachinery, Energy 285 (2023) 129405.

[247] P. Conti, M. Guo, A. Manzoni, A. Frangi, S. L. Brunton, J. Nathan Kutz, Multi-fidelity reduced-order surrogate modelling, Proceedings of the Royal Society A 480 (2283) (2024) 20230655.

[248] N. Demo, M. Tezzele, G. Rozza, A DeepONet multi-fidelity approach for residual learning in reduced order modeling, arXiv preprint arXiv:2302.12682 (2023).

[249] N. Geneva, N. Zabaras, Multi-fidelity generative deep learning turbulent flows, Foundations of Data Science 2 (4) (Mon Nov 30 19:00:00 EST 2020) 391–428. `doi:10.3934/fods.2020019`.

[250] S. Pawar, O. San, A. Nair, A. Rasheed, T. Kvamsdal, Model fusion with physics-guided machine learning: Projection-based reduced-order modeling, Physics of Fluids 33 (6) (2021).

[251] O. Sen, N. J. Gaul, K. Choi, G. Jacobs, H. Udaykumar, Evaluation of multifidelity surrogate modeling techniques to construct closure laws for drag in shock–particle interactions, Journal of Computational Physics 371 (2018) 434–451.

[252] M. Giselle Fernández-Godino, C. Park, N. H. Kim, R. T. Haftka, Issues in deciding whether to use multifidelity surrogates, AIAA Journal 57 (5) (2019) 2039–2054.

[253] G. A. Pavliotis, A. M. Stuart, Multiscale Methods, Vol. 53 of Texts Applied in Mathematics, Springer New York, New York, NY, 2008. `doi:10.1007/978-0-387-73829-1`.

[254] P. Constantin, C. Foias, B. Nicolaenko, R. Temam, Integral manifolds and inertial manifolds for dissipative partial differential equations, Vol. 70, Springer Science & Business Media, 1989.

[255] J. A. Sanders, F. Verhulst, J. Murdock, Averaging methods in nonlinear dynamical systems, Vol. 59, Springer, 2007.

[256] C. W. Gear, I. G. Kevrekidis, Projective methods for stiff differential equations: problems with gaps in their eigenvalue spectrum, SIAM Journal on Scientific Computing 24 (4) (2003) 1091–1106.

[257] A. Abdulle, E. Weinan, B. Engquist, E. Vanden-Eijnden, The heterogeneous multiscale method, Acta Numerica 21 (2012) 1–87.

[258] A. J. Majda, A. L. Bertozzi, Vorticity and Incompressible Flow, 1st Edition, Cambridge University Press, 2001. `doi:10.1017/CBO9780511613203`.

[259] H. Arbabi, J. E. Bunder, G. Samaey, A. J. Roberts, I. G. Kevrekidis, Linking machine learning with multiscale numerics: Data-driven discovery of homogenized equations, Jom 72 (2020) 4444–4457.

[260] J. L. Callaham, J.-C. Loiseau, S. L. Brunton, Multiscale model reduction for incompressible flows, Journal of Fluid Mechanics 973 (2023) A3. `doi:10.1017/jfm.2023.510`.

[261] C. J. Dsilva, R. Talmon, C. W. Gear, R. R. Coifman, I. G. Kevrekidis, Data-driven reduction for a class of multiscale fast-slow stochastic dynamical systems, SIAM Journal on Applied Dynamical Systems 15 (3) (2016) 1327–1351.

[262] E. R. Crabtree, J. M. Bello-Rivas, A. L. Ferguson, I. G. Kevrekidis, Gans and closures: Micro-macro consistency in multiscale modeling, Multiscale Modeling & Simulation 21 (3) (2023) 1122–1146.

[263] A. Liu, J. Axås, G. Haller, Data-driven modeling and forecasting of chaotic dynamics on inertial manifolds constructed as spectral submanifolds, arXiv preprint arXiv:2310.06067 (2023).

[264] C. Mou, B. Koc, O. San, L. G. Rebholz, T. Iliescu, Data-driven variational multiscale reduced order models, Computer Methods in Applied Mechanics and Engineering 373 (2021) 113470.

[265] O. San, S. Pawar, A. Rasheed, Variational multiscale reinforcement learning for discovering reduced order closure models of nonlinear spatiotemporal transport systems, Scientific Reports 12 (1) (2022) 17947.

[266] B. Koc, C. Mou, H. Liu, Z. Wang, G. Rozza, T. Iliescu, Verifiability of the data-driven variational multiscale reduced order model, Journal of Scientific Computing 93 (2) (2022) 54.

[267] C. Mou, N. Chen, T. Iliescu, An efficient data-driven multiscale stochastic reduced order modeling framework for complex systems, Journal of Computational Physics 493 (2023) 112450.

[268] S. Lee, Y. M. Psarellis, C. I. Siettos, I. G. Kevrekidis, Learning black-and gray-box chemotactic pdes/closures from agent based monte carlo simulation data, Journal of Mathematical Biology 87 (1) (2023) 15.

[269] P. Spalart, An Old-Fashioned Framework for Machine Learning in Turbulence Modeling (Aug. 2023). `arXiv: 2308.00837`, `doi:10.48550/arXiv.2308.00837`.

[270] M. Schmelzer, R. P. Dwight, P. Cinnella, Discovery of Algebraic Reynolds-Stress Models Using Sparse Symbolic Regression, Flow, Turbulence and Combustion 104 (2) (2020) 579–603. `doi:10.1007/s10494-019-00089-x`.

[271] L. Zanna, T. Bolton, Data-Driven Equation Discovery of Ocean Mesoscale Closures, Geophysical Research Letters 47 (17) (2020) e2020GL088376. `doi:10.1029/2020GL088376`.

[272] R. Mojgani, A. Chattopadhyay, P. Hassanzadeh, Discovery of interpretable structural model errors by combining Bayesian sparse regression and data assimilation: A chaotic Kuramoto–Sivashinsky test case, Chaos: An Interdisciplinary Journal of Nonlinear Science 32 (6) (2022) 061105. `doi:10.1063/5.0091282`.

[273] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. Hunt, M. Girvan, E. Ott, Hybrid Forecasting of Chaotic Processes: Using Machine Learning in Conjunction with a Knowledge-Based Model, Chaos: An Interdisciplinary Journal of Nonlinear Science 28 (4) (2018) 041101. `arXiv:1803.04779`, `doi:10.1063/1.5028373`.

[274] P. R. Vlachas, W. Byeon, Z. Y. Wan, T. P. Sapsis, P. Koumoutsakos, Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks, Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences 474 (2213) (2018) 20170844. `doi:10.1098/rspa.2017.0844`.

[275] H. Fan, J. Jiang, C. Zhang, X. Wang, Y.-C. Lai, Long-term prediction of chaotic systems with machine learning, Physical Review Research 2 (1) (2020) 012080. `doi:10.1103/PhysRevResearch.2.012080`.

[276] Z. Li, M. Liu-Schiaffini, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Learning Chaotic Dynamics in Dissipative Systems, Advances in Neural Information Processing Systems 35 (2022) 16768–16781.

[277] Y. Schiff, Z. Y. Wan, J. B. Parker, S. Hoyer, V. Kuleshov, F. Sha, L. Zepeda-Núñez, DySLIM: Dynamics Stable Learning by Invariant Measure for Chaotic Systems (Feb. 2024). `arXiv:2402.04467`, `doi:10.48550/arXiv.2402.04467`.

[278] G. Arranz, Y. Ling, S. Costa, K. Goc, A. Lozano-Duran, Building-block flow model for computational fluids (Mar. 2024). `arXiv:2403.09000`, `doi:10.48550/arXiv.2403.09000`.

[279] D. E. Keyes, L. C. McInnes, C. Woodward, W. Gropp, E. Myra, M. Pernice, J. Bell, J. Brown, A. Clo, J. Connors, et al., Multiphysics simulations: Challenges and opportunities, The International Journal of High Performance Computing Applications 27 (1) (2013) 4–83.

[280] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, P. Battaglia, Learning to Simulate Complex Physics with Graph Networks, in: Proceedings of the 37th International Conference on Machine Learning, PMLR, 2020, pp. 8459–8468.

[281] S. Taghizadeh, F. D. Witherden, Y. A. Hassan, S. S. Girimaji, Turbulence closure modeling with data-driven techniques: Investigation of generalizable deep neural networks, Physics of Fluids 33 (11) (2021) 115132. `doi:10.1063/5.0070890`.

[282] K. Kanov, R. Burns, C. Lalescu, G. Eyink, The Johns Hopkins turbulence databases: An open simulation laboratory for turbulence research, Computing in Science & Engineering 17 (5) (2015) 10–17.

[283] S. Rasp, P. D. Dueben, S. Scher, J. A. Weyn, S. Mouatadid, N. Thuerey, WeatherBench: a benchmark data set for data-driven weather forecasting, Journal of Advances in Modeling Earth Systems 12 (11) (2020) e2020MS002203.

[284] Z. Shen, A. Sridhar, Z. Tan, A. Jaruga, T. Schneider, A library of large-eddy simulations for calibrating cloud parameterizations, Authorea Preprints (2022).

[285] M. Takamoto, T. Praditia, R. Leiteritz, D. MacKinlay, F. Alesiani, D. Pflüger, M. Niepert, PDEBENCH: An Extensive Benchmark for Scientific Machine Learning (Mar. 2023). `arXiv:2210.07182`, `doi:10.48550/arXiv.2210.07182`.

[286] A. Towne, S. T. Dawson, G. A. Brès, A. Lozano-Durán, T. Saxton-Fox, A. Parthasarathy, A. R. Jones, H. Biler, C.-A. Yeh, H. D. Patel, et al., A database for reduced-complexity modeling of fluid flows, AIAA Journal (2023) 1–26.

[287] S. Yu, W. Hannah, L. Peng, J. Lin, M. A. Bhouri, R. Gupta, B. Lütjens, J. C. Will, G. Behrens, J. Busecke, et al., ClimSim: A large multi-scale dataset for hybrid physics-ML climate emulation, Advances in Neural Information Processing Systems 36 (2024).

[288] J. Sirignano, J. F. MacArt, J. B. Freund, DPM: A deep learning PDE augmentation method with application to large-eddy simulation, Journal of Computational Physics 423 (2020) 109811.

[289] H. Zhao, R. T. D. Combes, K. Zhang, G. Gordon, On learning invariant representations for domain adaptation, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, Vol. 97 of Proceedings of Machine Learning Research, PMLR, 2019, pp. 7523–7532.
URL https://proceedings.mlr.press/v97/zhao19a.html

Scientific Computing Group, Centrum Wiskunde & Informatica, Science Park 123, Amsterdam, The Netherlands
Email address: b.sanderse@cwi.nl

Advanced Computing, Mathematics and Data Division, Pacific Northwest National Laboratory, Richland, 99352, WA, USA
Email address: panagiotis.stinis@pnnl.gov

College of Information Sciences and Technology, The Pennsylvania State University, Westgate Building, University Park, 16802, PA, USA
Email address: rmaulik@psu.edu

Advanced Computing, Mathematics and Data Division, Pacific Northwest National Laboratory, Richland, 99352, WA, USA
Email address: shady.ahmed@okstate.edu