Ensemble-based learning of turbulence model from indirect observation data

Xin-Lei Zhang\textsuperscript{1,2}, Heng Xiao\textsuperscript{3†}, Xiaodong Luo\textsuperscript{4} and Guowei He\textsuperscript{1,2‡}

\textsuperscript{1}The State Key Laboratory of Nonlinear Mechanics, Institute of Mechanics, Chinese Academy of Sciences, Beijing 100049, China
\textsuperscript{2}School of Engineering Sciences, University of Chinese Academy of Sciences, Beijing 100049, China
\textsuperscript{3}Kevin T. Crofton Department of Aerospace and Ocean Engineering, Virginia Tech, Blacksburg, VA 24060, USA
\textsuperscript{4}Norwegian Research Centre (NORCE), Bergen, Norway

(Received xx; revised xx; accepted xx)

In this work, we propose using an ensemble Kalman method to learn a nonlinear eddy viscosity model, represented as a tensor basis neural network, from velocity data. Data-driven turbulence models have emerged as a promising alternative to traditional models for providing closure mapping from the mean velocities to Reynolds stresses. Most data-driven models in this category need full-field Reynolds stress data for training, which not only places stringent demand on the data generation but also makes the trained model ill-conditioned and lacks robustness. This difficulty can be alleviated by incorporating the Reynolds-averaged Navier-Stokes (RANS) solver in the training process. However, this would necessitate developing adjoint solvers of the RANS model, which can be challenging. Given such difficulty, we present an ensemble Kalman method with adaptive step size to train a neural network-based turbulence model by using indirect observation data. To our knowledge, this is the first such attempt in turbulence modelling. The ensemble method is first verified on the flow in a square duct, where it correctly learns the underlying turbulence models from velocity data. Then, the generalizability of the learned model is evaluated on a family of separated flows over periodic hills. It is demonstrated that the turbulence model learned in one flow can predict similar flows in different geometries.

1. Introduction

Despite the growth of available computational resources and the development of high-fidelity methods, industrial computational fluid dynamics (CFD) simulations still predominantly rely on Reynolds-averaged Navier-Stokes (RANS) solvers with turbulence models. This is expected to remain so in the decades to come, particularly for outer loop applications such as design optimization and uncertainty quantification (Slotnick \textit{et al.} 2014). Therefore, it is still of practical interest to develop more accurate and robust turbulence models.

Most of the currently used models are linear eddy viscosity models such as $k$–$\varepsilon$ model (Launder & Sharma 1974) and Spalart–Allmaras model (Spalart & Allmaras 1992), which are based on two major assumptions (Pope 2000): (1) weak equilibrium assumption, i.e., only the non-equilibrium in the magnitude of the Reynolds stress is accounted for through the transport equations, while its anisotropy is modelled based on

\[†\text{ Email address for correspondence: hengxiao@vt.edu}\]
\[‡\text{ Email address for correspondence: hgw@lnm.imech.ac.cn}\]
local strain rate, and (2) Boussinesq assumption, i.e., the Reynolds stress anisotropy is assumed to be aligned with the strain rate tensor. Reynolds stress transport models (also referred to as differential stress models) have been developed in the past few decades to address the shortcomings caused by the weak equilibrium assumption (Launder et al. 1975; Speziale et al. 1991; Eisfeld et al. 2016). As to the second assumption, various nonlinear eddy viscosity and explicit algebraic stress models have been developed (Spalart 2000; Wallin & Johansson 2000), and some have even achieved dramatic successes in specialized flows (e.g., those with secondary flows or rotation). However, these complex models face challenges from the lack of robustness, increased computational costs and implementation complexity, and the difficulty to generalize to a broader range of flows. Consequently, turbulence modellers and CFD practitioners often face a compromise between the predictive performance and practical usability (Xiao & Cinnella 2019).

In the past few years, data-driven methods have emerged as a promising alternative for developing more generalizable and robust turbulence models. For example, nonlocal models based on vector-cloud neural networks have been proposed to emulate Reynolds stress transport equations (Han et al. 2022; Zhou et al. 2022). While this line of research is still in an early stage, it has the potential of leading to more robust and flexible non-equilibrium Reynolds stress models without solving the tensorial transport equations. On the other hand, data-driven nonlinear eddy viscosity models have achieved much more success. Researchers have used machine learning to discover data-driven turbulence models or corrections thereto, which are nonlinear mappings from the strain rate and rotation rate to Reynolds stresses learned from data. Such functional mappings can be in the form of symbolic expressions (Weatheritt & Sandberg 2016; Schmelzer et al. 2020), tensor basis neural networks (Ling et al. 2016), and random forests (Wang et al. 2017; Wu et al. 2019a), among others. The data-driven nonlinear eddy viscosity models are a major improvement over their traditional counterparts in that they can leverage calibration data more systematically and explore a much larger functional space of stress–strain-rate mappings. However, they have some major shortcomings. First, as with their traditional counterparts, these data-driven models only addressed the Boussinesq assumption of the linear models as their strain–stress relations are still local, and thus they cannot address the weak equilibrium assumption described above. This is in contrast to the data-driven nonlocal Reynolds stress models (Han et al. 2022; Zhou et al. 2022), which emulates the Reynolds stress transport equations and fully non-equilibrium models. Second, the training of such models often requires full-field Reynolds stresses (referred to as direct data hereafter), which are rarely available except from high fidelity simulations such as direct numerical simulations (DNS) and wall-resolved large eddy simulations (LES) (Yang & Griffin 2021). This would inevitably constrain the training flows to those accessible for DNS and LES, i.e., flows with simple configurations at low Reynolds numbers. It is not clear that the data-driven models trained with such data would be applicable to practical industrial flows. Finally, the training of data-driven models is often performed in an a priori manner, i.e., without involving RANS solvers in the training process. Consequently, the trained model may have poor predictions of the mean velocity in a posteriori tests where the trained turbulence model is coupled with the RANS solvers. This is caused by the inconsistency between the training and prediction environments (Duraisamy 2021). Specifically, even small errors in the Reynolds stress can be dramatically amplified in the predicted velocities due to the intrinsic ill-conditioning of the RANS operator (Wu et al. 2019b; Brener et al. 2021). Such ill-conditioning is particularly prominent in high Reynolds number flows; even such apparently simple flow as a plane channel flow can be extremely ill-conditioned (Wu et al. 2019b). On the other hand, the model with the best a posterior performance may not necessarily excel in a priori evaluations (Park &
Ensemble-based learning of turbulence model

Choi 2021). In view of the drawbacks in a priori training of turbulence models with direct (Reynolds stress) data, it is desirable to leverage indirect observation data (e.g., sparse velocities and drag) to train data-driven turbulence models in the prediction environments by involving the RANS solvers in the training process. These indirect data are often available from experiments at high Reynolds numbers. Such a strategy is referred to as “model-consistent learning” in the literature (Duraisamy 2021).

Model-consistent learning amounts to finding the turbulence model that, when embedded in the RANS solvers, produces outputs in the best agreement with the training data. Specifically, in incompressible flows these outputs include the velocity and pressure as well as their post-processed or sparsely observed quantities. Assuming the turbulence model is represented with a neural network to be trained with the stochastic gradient descent method, every iteration in the training process involves solving the RANS equations and finding the sensitivity of the discrepancy between the observed and predicted velocities with respect to the neural network weights. This is in stark contrast to the traditional method of training neural networks that learns from direct data (output of the neural network, i.e., Reynolds stresses in this case), where the gradients can be directly obtained from back-propagation. In model-consistent training, one must typically resort to adjoint solvers to obtain the RANS solver-contributed gradient (sensitivity of velocity with respect to Reynolds stresses), as the full model consists of both the neural network and the RANS solver (Holland et al. 2019; Michelén Ströfer & Xiao 2021). The adjoint sensitivity is then multiplied to the neural network gradient according to the chain rule to yield the full gradient. Similar efforts of combining adjoint solvers and neural network gradient have been made in learning sub-grid-scale models in LES (MacArt et al. 2021).

Moreover, for turbulence models represented as symbolic expressions, model-consistent learning is similarly performed by combining the model with the RANS solver in the learning processes (Zhao et al. 2020), although the chain-rule based gradient evaluation is no longer needed in gradient-free optimizations such as genetic optimization.

In view of the difficulty in developing adjoint solvers (particularly for legacy codes and multi-physics coupled solvers), Michelén Ströfer et al. (2021b) explored ensemble-based gradient approximation as an alternative to the adjoint solver used in Michelén Ströfer & Xiao (2021) to learn turbulence model from indirect data. Such a gradient is combined with that from the neural network via chain rule and then used in an explicit gradient-descent training. They found that the learned model was less accurate than that learned by using adjoint solvers. This is not surprising, because the ensemble-based gradient approximation is less accurate than the analytic gradient from the adjoint solvers (Evensen 2018). Therefore, instead of using an ensemble to approximate gradients in optimization, it can be advantageous to directly use ensemble Kalman methods for training neural networks (Chen et al. 2019; Kovachki & Stuart 2019). This is because such ensemble methods do not merely perform explicit, first-order gradient-descent optimization as is typically done in neural network training (deep learning). Rather, they implicitly use the Hessian matrix (second-order gradient) along with the Jacobian (first-order gradient) to accelerate convergence. Indeed, ensemble-based learning has gained significant success recently (Schneider et al. 2020a,b), but the applications focused mostly on learning from direct data. They have not been used to learn from indirect data, where physical models such as RANS solvers become an integral part of the learning process.

In this work, we propose using an iterative ensemble Kalman method to train a neural network-based turbulence model by using indirect observation data. To the authors’ knowledge, this is the first such attempt in turbulence modelling. Moreover, in view of the strong nonlinearity of the problem, we adjust the step size adaptively in the learning process (Luo et al. 2015), which serves a similar purpose to that of the learning-rate
scheduling in deep learning. Such an algorithmic modification is crucial for accelerating convergence and improving robustness of the learning, which can make an otherwise intractable learning problem with the adjoint method (Michelén Ströfer & Xiao 2021) computationally feasible with the ensemble method. We show that, by incorporating Hessian information with adaptive stepping, the ensemble Kalman method exceeds the performance of the adjoint-based learning (Michelén Ströfer & Xiao 2021) in both accuracy and robustness. Specifically, the present method successfully learned a generalizable nonlinear eddy viscosity model for the separated flows over periodic hills (Section 4), which the adjoint method was not able to achieve due to the lack of robustness. We emphasize that all these improvements are achieved at a much lower computational cost (measured in wall-time) and with a significantly lower implementation effort compared to the adjoint method (Michelén Ströfer & Xiao 2021). Both methods used the same representation of Reynolds stresses based on the tensor basis neural network (Ling et al. 2016).

In summary, the present framework of ensemble-based learning from indirect data has three key advantages. First, compared to methods that learn from direct data, the present framework relaxes the data requirements and only needs the measurable flow quantities, e.g., sparse measurements of the mean velocities or integral quantities such as drag and lift, rather than full-field Reynolds stresses. Second, the model is trained in the prediction environment, thereby alleviating the ill-condition of the explicit data-driven RANS equation and avoiding the inconsistency between training and prediction. Finally, the ensemble method is non-intrusive and thus very straightforward to implement for any solvers. In particular, it does not require adjoint solvers, which allows different quantities to be used in the objective function without additional development.

The rest of this paper is organized as follows. The architecture of the neural network and the model-consistent training algorithm are presented in Section 2. The case setup for testing the performance of the proposed non-intrusive model-consistent training workflow is detailed in Section 3. The training results are presented and analyzed in Section 4. The parallelization and the flexibility of the proposed method are discussed in Section 5. Finally, conclusions are provided in Section 6.

2. Reynolds stress representation and model-consistent training

The objective is to develop a data-driven turbulence modelling framework that meets the following requirements:

(i) The Reynolds stress representation shall be frame invariant and sufficiently flexible in expressive power, which enables it to become a universal model trained on a wide range of flows.

(ii) The model shall be trained in the prediction environment for robustness.

(iii) It shall be able to incorporate sparse and potentially noisy observation data as well as Reynolds stress data.

To this end, we choose the tensor basis neural networks (Ling et al. 2016) to represent the mapping from the mean velocities to the Reynolds stresses. Furthermore, we use the ensemble Kalman method to learn the neural network-based model in a non-intrusive, model-consistent manner.

The proposed workflow for training the tensor basis neural networks with indirect observation data is schematically illustrated in Figure 1. Traditionally, ensemble Kalman methods have been used in data assimilation applications to infer the state of the system (e.g., velocities and pressures of a flow field). However, in our application, we aim to learn a turbulence model represented by a neural network. Therefore, the parameters (weight
Figure 1: Schematic of the ensemble-based learning with sparse velocity data, consisting of the following four steps: (a) sampling the weights of the tensor basis neural network; (b) construct the Reynolds stress by evaluating neural network-based turbulence model; (c) propagate the constructed Reynolds stress tensor to velocity by solving RANS equations; (d) update the neural network weights by incorporating observation data.

vector $\mathbf{w}$ of the network are the quantities to be inferred. The iterative ensemble Kalman method adopted for model learning consists of the following steps:

(i) Sample the parameters (neural network weight vector $\mathbf{w}$) based on the initial prior distribution (Fig. 1a).

(ii) Construct Reynolds stress field from the mean velocity field by evaluating the neural network-based turbulence model (Fig. 1b). For a given mean velocity field $\mathbf{u}(\mathbf{x})$, each of the sample $\mathbf{w}_j$ (with $j$ being sample index) implies a different turbulence model and thus a different Reynolds stress field, leading to an ensemble of Reynolds stress field;

(iii) Propagate each Reynolds stress field in the ensemble to velocity field by solving the RANS equations (Fig. 1c), based on which the observations can be obtained via post-processing (e.g., extracting velocities at specific points or integrating surface pressure to obtain drag);

(iv) Update the parameters (network weights $\mathbf{w}$) through statistical analysis of the predicted observable quantities (e.g., velocities or drag) and comparison to observation data (Fig. 1d).

Steps (ii)–(iv) are repeated until convergence is achieved. The implementation details are provided in Appendix A.

In this section, we introduce the Reynolds stress representation based on tensor basis neural network and the ensemble-based learning algorithm. The latter is compared to other learning algorithms in the literature.
2.1. Embedded neural network for Reynolds stress representation

For constant-density, incompressible turbulent flows, the mean flow can be described by the RANS equation:

\[ \nabla \cdot u = 0 \]
\[ u \cdot \nabla u = -\nabla p + \nu \nabla^2 u - \nabla \cdot \tau, \]

(2.1)

where \( p \) denotes mean pressure normalized by the constant flow density, and the Reynolds stress \( \tau \) indicates the effects of the small-scale turbulence on the mean flow quantities, which are required to be modelled. The Reynolds stress can be decomposed into an anisotropic part \( a \) and an isotropic part as

\[ \tau = a + \frac{2}{3} k I, \]

(2.2)

where \( k \) is the turbulence kinetic energy, and \( I \) is the second order identity tensor. Different strategies have been developed to represent the anisotropic part of the Reynolds stress, and here we use the tensor basis neural network (Ling et al. 2016).

The neural network represents the anisotropic part of Reynolds stress with the scalar invariants and the tensor bases of turbulence field. Specifically, the neural network is used to represent the mapping between the scalar invariants and coefficients of tensor bases. Further, the output of the neural network is combined with the tensor bases to construct the Reynolds stress field such that the framework has the embedded Galilean invariance. The anisotropic part of the Reynolds stress \( a \) can be constructed as (Pope 1975):

\[ a = 2k \sum_{i=1}^{10} g^{(i)} T^{(i)}, \]

(2.3)

with \( g^{(i)} = g^{(i)}(\theta_1, \ldots, \theta_5) \),

(2.4)

where \( T \) and \( \theta \) are the tensor basis and scalar invariant of the input tensors, and \( g \) is the scalar coefficient functions to be learned. There are 10 independent tensors that give the most general form of eddy viscosity. The first four tensors are given as

\[ T^{(1)} = S, \quad T^{(2)} = SW - WS \]
\[ T^{(3)} = S^2 - \frac{1}{3} \{S^2\} I, \quad T^{(4)} = W^2 - \frac{1}{3} \{W^2\} I \]

(2.5)

where the curly bracket \( \{\cdot\} \) indicates the trace of a matrix. The first two scalar invariants are

\[ \theta_1 = \{S^2\} \quad \text{and} \quad \theta_2 = \{W^2\}. \]

(2.6)

Both the symmetric tensor \( S \) and the anti-symmetric tensor \( W \) are normalized by the turbulence time scale \( \frac{k}{\varepsilon} \) as \( S = \frac{k}{2 \varepsilon} [\nabla u + (\nabla u)^\top] \) and \( W = \frac{k}{2 \varepsilon} [\nabla u - (\nabla u)^\top] \). The time scale \( \frac{k}{\varepsilon} \) is obtained from the turbulent quantities solved from the transport equations for turbulence kinetic energy \( k \) and dissipation rate \( \varepsilon \). For a two-dimensional flow, only two scalar invariants are nonzero, and the first three tensor bases are linearly independent (Pope 1975). Further for incompressible flow, the third tensor basis can be incorporated into the pressure term in the RANS equation, leaving only two tensor functions and two scalar invariants. In the turbulence transport equation, the turbulence production term is modified to account for the expanded formulation of Reynolds stress \( P = -\tau : S \), where \( : \) denotes double contraction of tensors. For details of the implementation, readers are referred to Michélen Ströfer & Xiao (2021). Note that the representation...
Ensemble-based learning of turbulence model

of the Reynolds stress is based on the following three hypotheses: (1) the Reynolds stress can be locally described with the scalar invariant and the independent tensors; (2) the projection of the Reynolds stress onto the tensor bases can be represented by a neural network; (3) a universal model form exists for flows with similar features.

In this work the tensor basis neural network is embedded into the RANS equation during the training process. Specifically, the RANS equation is solved to provide velocity prediction by coupling with the neural network-based model, and the predicted velocity is used to train the neural network by using the indirect observations, e.g., velocities. We use an ensemble Kalman method to train the neural network-based turbulence model embedded in the RANS equations, which is shown in Section 2.2 below. More detailed comparison between the proposed method and other related schemes are presented in Section 2.3.

2.2. Ensemble-based model-consistent training

The goal of the model-consistent training is to reduce the model prediction error by optimizing the weights $w$ of the neural network. The corresponding cost function can be formulated as

$$ J = \|w - w^0\|^2_P + \|y - \mathcal{H}[w]\|^2_{\gamma R}, \quad (2.7) $$

where $\| \cdot \|_A$ indicates weighted norm (defined as $\|v\|^2_A = v^T A^{-1} v$ for a vector $v$ with weight matrix $A$), $P$ is the model error covariance matrix indicating the uncertainties on the initial weights, $R$ is the observation error covariance matrix, $\gamma$ is a scaling parameter, and $y$ is the training data which is subjected to the Gaussian noise $\epsilon \sim N(0, R)$. For simplicity we introduce the operator $\mathcal{H}$, which is a composition of RANS solver and the associated post-processing (observation). It maps the weights $w$ to the observation space (e.g., velocity or drag coefficient). The first term in Equation (2.7) is introduced to regularize the updated weights $w$ by penalizing large departure from their initial values $w^0$. The second term describes the discrepancy between the model prediction $\mathcal{H}[w]$ and the observation $y$. The training of the neural network is equivalent to minimization of the cost function (2.7) by optimizing the weights $w$. Note that the cost function can be modified to include other observation quantities such as friction coefficient and transition location.

In this work, we use the iterative ensemble Kalman method with adaptive stepping (Luo et al. 2015) to train the neural network framework. This algorithm is a variant of the ensemble-based method where the observation error covariance matrix $R$ is inflated such that the step size is adjusted adaptively at each iteration step. The weight update scheme of the iterative ensemble Kalman method is formulated as

$$ w_{j+1} = w_j + K (y_j - \mathcal{H}[w_j]), \quad (2.8a) $$

with

$$ K = S_w S_y^T (S_y S_y^T + \gamma^4 R)^{-1}, \quad (2.8b) $$

where $l$ is the iteration index and $j$ is the sample index. The square root matrices $S_w$ and $S_y$ can be estimated from the ensemble at each iteration. See step (vi) and Equation (A 1) of the detailed implementation in Appendix A.

Note that the Kalman gain matrix above has a slightly different form than the more common formulation $K = PH^T (HPH^T + \gamma^4 R)^{-1}$. This is because we have written the terms associated with the prediction error covariance matrix $P$ by using the square root matrix $S_w$ and its projection $S_y$ to the observation space, i.e.,

$$ P = S_w S_w^T \quad \text{and} \quad S_y = HS_w \quad (2.9) $$
where $H$ is the local gradient of the observation operator $\mathcal{H}$ with respect to the parameter $w$. The equivalence between the two formulations is illustrated in Appendix B.

The Kalman gain matrix in Equation (2.8b) implicitly contains the inverse of the approximated second-order derivatives (Hessian matrix) as well as the gradient (Jacobian) of the cost function (both with respect to the weights $w$). This can be seen from the derivations presented in Appendix B. Including both the gradient and the Hessian information significantly accelerate the convergence of the iteration process and thus improves the learning efficiency. This is in stark contrast to using only the gradient in typical training procedures of deep learning. Moreover, this is done in ensemble Kalman methods economically without significant overhead in computational costs or memory footprint.

The inflation parameter $\gamma^l$ in Equation (2.8b) can be considered a coefficient for adjusting the relative weight between the prediction discrepancies and the regularization terms. As such, we let

$$
\gamma^l = \beta^l \{S^l_y (S^l_y)^\top\} / \{R\},
$$

where $\beta^l$ is a scalar coefficient whose value also changes over the iteration process. The detailed algorithm for scheduling $\beta^l$ (and thus $\gamma^l$) is presented in step (vii) of the detailed implementation in Appendix A.

The ensemble-based method has the following three practical advantages. First, it produces an ensemble of weights of the neural network, based on which uncertainty quantification can be conducted for the model prediction similarly to the Bayesian neural network (Sun & Wang 2020). Second, unlike the adjoint-based method, the ensemble-based method is non-intrusive and derivative-free, which means that it can be applied to black-box systems without the need for modifying the underlying source code. This feature makes it convenient to implement the ensemble-based method in practice and promotes the generalizability of the implemented ensemble method to different problems. Finally, to reduce the consumption of computer memory, commonly used training algorithms, such as stochastic gradient descent, typically only involve the use of gradients of an objective function to update the weights of a neural network, while the ensemble-based method incorporates the information of low-rank approximated Hessian without a substantial increment of computer memory. Utilizing the Hessian information significantly improves convergence as discussed above. In addition, the method can be used to train the model jointly with data from different flow configurations. In such scenarios, the observation vector and the corresponding error covariance matrix would contain different quantities, e.g., the velocity and drag coefficient.

The open-source code OpenFOAM (The OpenFOAM Foundation 2021) is used in this work to solve the RANS equations with turbulence models. Specifically, the built-in solver *simpleFoam* is applied to solve the RANS equation coupling with the specialized neural network model. Moreover, the DAFI code (Michelén Ströfer et al. 2021a) is used to implement the ensemble-based training algorithm. A fully connected neural network is used in this work, and the detailed architecture for each case will be explained later. The rectified linear unit (ReLU) activation function is used for the hidden layers, and the linear activation function is used for the output layer. The machine learning library TensorFlow (Abadi et al. 2015) is employed to construct the neural network. The code developed for this work is publicly available on Github (Zhang et al. 2022).

2.3. Comparison to other learning methods

Various approaches have been proposed for data-driven turbulence modelling, such as the direct training method (Ling et al. 2016), the adjoint-based differentiable
method (Michelén Ströfer & Xiao 2021), the ensemble gradient method (Michelén Ströfer et al. 2021b), and the ensemble Kalman inversion (Kovachki & Stuart 2019). Here we present an algorithmic comparison of the proposed method with other model learning strategies in a unified perspective.

Conventional methods use the Reynolds stress of DNS to train the model in the a priori manner, with the goal to minimize the discrepancy between the output of a neural network and the training data based on the back propagation technique. This concept can be formulated as a corresponding minimization problem (with the proposed solution), as follow:

$$
\arg\min_{w} J = \|\tau(w, \tilde{S}, \tilde{W}) - \tau^{DNS}\|^2, \\
w^{l+1} = w^l + \beta \frac{\partial \tau(w, \tilde{S}, \tilde{W})}{\partial w} \left[ \tau(w, \tilde{S}, \tilde{W}) - \tau^{DNS} \right],
$$

(2.10)

where the input features $\tilde{S}$ and $\tilde{W}$ are processed from the DNS results. Further the trained neural network is coupled with the RANS solver for the posterior tests in similar configurations. It is obvious that inconsistency exists between the training and prediction environments. Specifically, during the training process, the model inputs are post-processed from the DNS data, while the learned model uses the RANS prediction to construct the input features. On the other hand, the training process aims to minimize the cost function associated with the Reynolds stress, while the prediction aims to achieve the least discrepancies in the velocity. This inconsistency would lead to unsatisfactory prediction due to the ill-conditioning issue of the RANS equation (Wu et al. 2019b). To tackle this problem, model-consistent training is required to construct the input features and the cost function with respect to more appropriate predicted quantities, e.g., the velocity.

For model-consistent training, the corresponding minimization problem (together with its solution) is changed to

$$
\arg\min_{w} J = \|u^{DNS} - u(\tau(w, S, W))\|^2, \\
w^{l+1} = w^l + \beta \frac{\partial J}{\partial w},
$$

(2.11)

where the input feature $S$ and $W$ are processed from the RANS prediction. Both the input feature and the objective function used for training are consistent with the prediction environment. Different approaches can be used to train the model, such as the adjoint-based differentiable method, the ensemble-based gradient method, and the ensemble Kalman inversion method. Specifically, the adjoint-based differentiable framework (Michelén Ströfer et al. 2020) decomposes the gradient of the cost function into $\frac{\partial J}{\partial \tau}$ and $\frac{\partial \tau}{\partial w}$ by using the chain rule. The weight-update scheme can be written as

$$
w^{l+1} = w^l + \beta \frac{\partial J}{\partial \tau} \frac{\partial \tau}{\partial w}.
$$

(2.12)

The gradient $\frac{\partial J}{\partial \tau}$ is computed using the adjoint method, and the gradient $\frac{\partial \tau}{\partial w}$ is computed based on the back propagation method. The ensemble-based gradient method applies the Monte Carlo technique to draw samples from a Gaussian distribution. Moreover, the data noise is taken into account by weighting the cost function with the observation error covariance matrix $R$. Further, the cross-covariance matrix computed by the ensemble method can be used to approximate the adjoint-based gradient as

$$
\frac{\partial J}{\partial \tau} \approx S_w S_y^T R^{-1} (H[w] - y).
$$

(2.13)
The above-mentioned training approach employs the readily available analytic gradient of the neural network based on the back propagation method. Further the gradient of the cost function can be constructed by coupling with adjoint- or ensemble-based sensitivity of the RANS equation.

The ensemble Kalman inversion method (Kovachki & Stuart 2019) adds a regularization term into the cost function and approximates the gradient of the cost function with respect to the weights of the neural network based on implicit linearization. The minimization problem and the corresponding weight update scheme are

$$\arg \min_w J = \|w^{l+1} - w^l\|^2_p + \|u^\text{DNS} - u\|^2_R$$

$$w_j^{l+1} = w_j^l + S_w^l \left( S_y^l (S_y^l)^\top + R \right)^{-1} (y_j - \mathcal{H}[w^l]).$$

Note that this method involves the Hessian of the cost function (Evensen 2018; Luo 2021) and provides quantified uncertainties based on Bayesian analysis (Zhang et al. 2020). Similar to the ensemble gradient method, the ensemble Kalman inversion method also approximates the sensitivity of velocity to neural-network weights based on the ensemble cross-covariance matrix, without involving the analytic gradient of the neural network. However, the ensemble Kalman inversion method includes approximated Hessian in the weight-update scheme, which is missing in the ensemble gradient method. The present algorithm can be considered a variant of the ensemble Kalman inversion method, which inherits the advantages of ensemble-based methods in terms of the non-intrusiveness and quantified uncertainty. Moreover, the present method adjusts the relative weight of the prediction discrepancy and the regularization terms at each iteration step, which helps to speed up the convergence of the iteration process and enhance the robustness of the weight-update scheme. For convenience of comparison, the training algorithms of different model-consistent data-driven turbulence modelling frameworks are summarized in Table 1.

### 3. Case setup

We use two test cases to show the performance of the proposed method for learning turbulence models: (1) flow in a square duct, and (2) separated flows over periodic hills. Both are challenging classical test cases for linear eddy viscosity models. We aim to learn neural network-represented nonlinear eddy viscosity models from velocity data by using the ensemble method. The learned models are evaluated by comparing to the ground truth for the square duct case and assessing its generalization performance in the separated flows over periodic hills. The results are also compared to those of the adjoint-based method (Michelén Ströfer & Xiao 2021). Details of the case setup are discussed below.

#### 3.1. Secondary flows in a square duct

The first case is the flow in a square duct, where the linear eddy viscosity model is not able to capture the in-plane secondary flow. The nonlinear eddy viscosity model, e.g., Shih’s quadratic model (Shih 1993), is able to simulate the secondary flows. On the other hand, Shih’s quadratic model provides an explicit formula of the mapping between the scalar invariant $\theta$ and the function $g$. In Shih’s quadratic model, the $g$ function of the
Table 1: The ensemble Kalman method with adaptive stepping (Kovachki & Stuart 2019; Luo et al. 2015) for learning turbulence model from indirect data (e.g., DNS mean velocities). To view the method in a unified perspective, we compared its cost function and update scheme with other related methods, including learning from direct data (i.e., Reynolds stresses) (Ling et al. 2016), adjoint-based learning (Michelén Ströfer & Xiao 2021), and ensemble gradient learning (Michelén Ströfer et al. 2021b).

| Method                                      | Cost function | Update scheme |
|---------------------------------------------|---------------|---------------|
| Learning from direct data                   | \( J = \|\tau^{DNS} - \tau\|^2 \) | \( \tau^{l+1} = \tau^l + \beta \frac{\partial \tau}{\partial \omega} (\tau^{DNS} - \tau) \) |
| Adjoint-based learning                      | \( J = \|u^{DNS} - u\|^2 \) | \( u^{l+1} = u^l + \beta \frac{\partial J}{\partial \tau} \partial \tau \partial \tau \) |
| Ensemble gradient learning                 | \( J = \|u^{DNS} - u\|^2 \) | \( w^{l+1}_j = w^l_j + K(y_j - H[w_j]) \) |
| with adaptive stepping (present framework) | \( J = \|w^{l+1}_j - w_j\|^2 \) | \( K = S_w S_y^T R^{-1} \) |

Hence we use the velocity results from Shih’s quadratic model as the synthetic truth and show that the method is able to reveal the underlying relationship between the scalar invariant and the tensor basis. On the other hand, we aim to compare the adjoint-based and the present ensemble-based methods in terms of the training accuracy and efficiency in this case.

The flow in square duct is fully developed, and only one cell is used in the stream-wise direction. Moreover, one quarter of the domain is used due to the symmetry, and the mesh grid is 50 × 50. As for the architecture of the neural network in this case, two scalar invariants are used as input features, and four \( g \) functions \( g^{(1-4)} \) are used in the output layer. The input features of the synthetic truth are shown in Figure 2. Since the stream-wise velocity \( u_x \) is dominant, the first two scalar invariants are approximately equal in the magnitude but with opposite signs. The slight difference between the scalar invariants \( \theta_1 \) and \( \theta_2 \) is caused by the secondary flow in the plane. We also provide the plot of \( |\theta_1| - |\theta_2| \), which indicates the relative importance of the strain rate and the vorticity. The stream-wise velocity gradient is relatively small near the center of the duct, leading

\[
\begin{align*}
g^{(1)}(\theta_1, \theta_2) &= \frac{-2/3}{1.25 + \sqrt{2} \theta_1 + 0.9 \sqrt{-2} \theta_2}, \\
g^{(2)}(\theta_1, \theta_2) &= \frac{7.5}{1000 + (\sqrt{2} \theta_1)^3}, \\
g^{(3)}(\theta_1, \theta_2) &= \frac{1.5}{1000 + (\sqrt{2} \theta_1)^3}, \\
g^{(4)}(\theta_1, \theta_2) &= \frac{-9.5}{1000 + (\sqrt{2} \theta_1)^3}.
\end{align*}
\]
to the negligible scalar invariant $\theta_1$. Moreover, the shear strain rate is dominant near the duct center, while there is a pair of vortices indicating the strong rotation rate. Besides, it can be seen that the range of the input features is from 0 to approximately 7. We draw 50 samples of the neural network weights in this case. In the neural network, we use 2 hidden layers with 5 neurons per layer. A sensitivity study of the training algorithm to the neural network architecture and the observation data is provided in Appendix C.

3.2. Separated flow over periodic hills

The flow over periodic hills is a canonical separated flow for the numerical investigation of turbulence models. There is no ground truth for the model function which is able to capture the flow characteristics accurately. Here we use the DNS results (Xiao et al. 2020) as the training data and learn the neural network-based model by using the ensemble-based method. Further, we validate the generalizability of the learned model in similar configurations with varying slopes (Xiao et al. 2020). Specifically, the hill geometry is parameterized with the slope coefficient $\alpha$. The separation extent decreases as the slope $\alpha$ increases from 0.5 to 1.5. The case with slope parameter $\alpha = 1$ is used as the training case, and the cases with other slopes of $\alpha = 0.5, 0.8, 1.2, 1.5$ are used to test the generalizability of the learned model in the scenarios having different levels of flow separation. The mesh is set as 149 cells in stream-wise direction and 99 cells in normal direction after grid-independence tests.

For the two-dimensional incompressible flow, there are only the first two scalar invariants and independent tensors after merging the third tensor basis into the pressure term in the RANS equation (Michelén Ströfer & Xiao 2021). The input features of the DNS data are shown in Figure 2, scaled with RANS predicted time scale. The plot of the first scalar invariant $\theta_1$ indicates the large strain rate in the free shear layer and the windward side of the hill. The second scalar invariant $\theta_2$ shows the vorticity mainly in the flow separation region at the leeward side of the hill. From the plot of $|\theta_1| - |\theta_2|$, it can be seen that the magnitude of the first two scalars is equivalent in most areas. The strong vorticity in the downhill is caused by the flow separation, while near the uphill region the shear strain rate is dominant due to the channel contraction. Compared to the square duct case, the separated flow over periodic hills has a wider range in the magnitude of the input features, which is from 0 to about 100. That is because in the square duct case, the magnitude of the scalar invariant is mainly determined by the stream-wise velocity $u_x$, while in the periodic hill case, both $u_x$ and $u_y$ have considerable effects on the input features. Moreover, the magnitude of the time scale in the periodic hill is much larger than that in the square duct flow. Concretely, the maximum value for the periodic hill case is about 490, while that for the square duct case is about 10. Hence, we use a deeper neural network of 10 hidden layers with 10 neurons per layer compared to the square duct case. We draw 50 samples of the neural network weights in this case. The training data set is summarized in Table 2.

4. Results

4.1. Flow in a square duct: learning underlying closure functions

We first use the proposed ensemble-based method to train the turbulence model for flows in a square duct, and the results show that the predicted Reynolds stress has a good agreement with the synthetic ground truth (Equation 3.1). The plots of the in-plane velocity vector and the Reynolds stress are presented in Figure 3 with comparison to the adjoint-based method and the ground truth. The contour lines for $u_y$ are indicated
in the plot of velocity to clearly show similar patterns among the ground truth, the adjoint method, and the ensemble-based method. The contour plots of the Reynolds stress in $\tau_{xy}$ and $\tau_{yz}$ are used to demonstrate the ability of the ensemble method in discovering the underlying Reynolds stress model given velocity data. The in-plane velocity is driven by Reynolds normal stresses imbalance $\tau_{yy} - \tau_{zz}$, which is evident from the vorticity transport equation (Launder & Sandham 2002). As such, the imbalance $\tau_{yy} - \tau_{zz}$ is also presented in Figure 3, demonstrating that the Reynolds stress field is accurately learned from the in-plane velocities. The learned model with the proposed method achieves similar results in both the velocity and Reynolds stress to those of the adjoint-based method. It is noted that in this case the entire field is used as the training data. By using fewer observations, e.g., only velocity data on the anti-diagonal line (upper right corner to lower left corner), the full velocity field can be also recovered and the Reynolds stresses are correctly learned, but the errors are larger, especially in velocity. This is presented in Appendix C. The results demonstrate that the proposed method is able to learn the underlying turbulence model, which in turn provides good predictions of velocities and Reynolds stresses.

To clearly show the performance of the trained model, we provide the prediction error in the velocity and the Reynolds stress. The prediction error over the computational domain is defined as

$$\mathcal{E}(q) = \frac{\|q^{\text{predict}} - q^{\text{truth}}\|}{\|q^{\text{truth}}\|}. \quad (4.1)$$

The comparison between the adjoint and ensemble-based methods in prediction error of velocity and Reynolds stress as well as the training efficiency is provided in Table 3.
The results confirm that both adjoint and ensemble-based methods are able to achieve satisfactory agreement in the velocities and to predict the Reynolds stresses well. On the other hand, the adjoint-based method provides slightly better predictions than the ensemble method. Specifically, the errors in velocity and Reynolds stress with the adjoint-based method are 0.1% and 4.5%, respectively, while those for the ensemble method are 0.47% and 5.8%, respectively.

As for the training efficiency, the adjoint-based method is more time-consuming compared to the ensemble-based method as shown in Table 3. Specifically, the adjoint-based method requires approximately 1000 iterations which significantly increase the wall time to about 133 hours in this case. In contrast, the ensemble-based method is efficient to obtain comparable prediction results within 3.6 hours. To achieve the error reduction of $\mathcal{E}(u) < 0.005$, the adjoint method requires 238 steps and a wall time of 32 hours, while the ensemble-based method can reach the same error within only 0.6 hours. That is mostly due to the use of Hessian information and the covariance inflation factor $\gamma$, which dynamically adjusts the relative weight of the cost function to accelerate the convergence.

We further show the good reconstruction in the scalar invariant $\theta_1$ and $|\theta_1| - |\theta_2|$ with the ensemble-based method compared to the ground truth. The contour plots of

Figure 3: Reynolds shear stresses $\tau_{xy}$ and $\tau_{yz}$ (middle columns) and normal stresses imbalance $\tau_{yy} - \tau_{zz}$ (right column) in the square duct predicted from the models learned by the adjoint (center row) and ensemble method (bottom row), compared against the ground truth (top row). The velocities (left column) are indicated by vector plots along with contours of the in-plane velocity $u_y$ scaled by a factor of 1000.
| Method             | $\mathcal{E}(u)$ | $\mathcal{E}(\tau)$ | Total steps | Wall time  | Steps ($\mathcal{E}(u) < 0.005$) | Wall time ($\mathcal{E}(u) < 0.005$) |
|-------------------|-------------------|----------------------|-------------|------------|----------------------------------|-------------------------------------|
| Adjoint-based     | 0.1%              | 4.5%                 | 1000        | 133 hours  | 238                              | 32 hours                            |
| Ensemble-based    | 0.47%             | 5.8%                 | 50          | 3.6 hours  | 8                                | 0.6 hours                           |

Table 3: Comparison of the predictive error and time cost between adjoint-based and ensemble-based learning

![Image](image1)

(a) contours of invariants

![Image](image2)

(b) kernel density of $\theta_1$

Figure 4: (a) Comparison of scalar invariant $\theta_1$ and $|\theta_1| - |\theta_2|$ among the adjoint-based learned model, the ensemble-based learned model, and the truth; (b) Kernel density plot of $\theta_1$ from the truth and the prediction. The circle indicates the 30% quantile (i.e., 30% of the cells have $\theta_1$ smaller than this value). The probability densities of the truth and the prediction are plotted on the margins.

The scalar invariant are presented in Figure 4. The predicted scalar invariant with the learned model agrees well with the ground truth. The difference between the initial and the truth is mainly due to the in-plane secondary flow which cannot be captured by the linear eddy viscosity model. With the learned models, the flow field in the y-z plane is well predicted, which further improves the estimate of the scalar invariant. It is observed that slight differences exist near the duct center. In that region, there are mainly small values of the scalar invariant $\theta$, due to the negligible stream-wise velocity gradient. Additionally, we provide the predicted scalar invariant compared to the ground truth, which clearly shows the good agreements between the prediction and the truth. The probability density function (PDF) of the scalar invariant $\theta$ is also plotted in Figure 4, showing the significantly small probability for $\theta$ less than about 5. The 30% quantile is located approximately at 5.1, indicating that only 30% of the cells in the domain have $\theta_1$ smaller than this value.

The learned functional mapping between the scalar invariant $\theta$ and the tensor basis coefficient $g$ also have a good agreement with the ground truth. This is illustrated in Figure 5. Since the two invariants are linearly correlated ($\theta_1 \approx -\theta_2$) we only show the plot of the mapping from the scalar invariant $\theta_1$ to the coefficients $g$. In the duct flow, the in-plane velocity is affected by a linear combination $g^{(2)} - 0.5g^{(3)} + 0.5g^{(4)}$ of the
Figure 5: Comparison plots of the functional mapping between the scalar invariant and the tensor coefficient $g$ among the truth, the baseline $k$–$\varepsilon$ model, and the models learned with adjoint and ensemble methods.

$g$ functions. It can be seen that the learned mapping can have a good agreement with the ground truth (the $g(\theta)$ in Equation 3.1) implied by Shih’s quadratic model. We note that large differences exist in the region with small values of $\theta_1$, particularly for the combination $g^{(2)} - 0.5g^{(3)} + 0.5g^{(4)}$. That is because the velocity is affected by the product of the $g$ function and the tensor bases $T$. In the region with small $\theta_1$ (near the center of the duct), the magnitudes of the tensor bases $T^{(1)}$ and $T^{(2)}$ (even after normalization with $k/\varepsilon$) are small, and thus the velocities are no longer sensitive to the $g$ functions. Moreover, small values of $\theta_1$ are represented by only a small number of cells in the domain, which is evident from Figure 4(b). This lack of representation makes it difficult to learn the underlying mapping in the region with small $\theta_1$. However, we note that the ensemble method achieves qualitatively similar results (albeit with errors of opposite signs) with the adjoint-based method in the functional mapping. This suggests that the bottleneck for learning the complete mapping lies in the intrinsic ill-conditioning of the problem (insensitivity to small $\theta_1$ magnitudes) rather than the lack of analytic gradient. This can be remedied by learning from several flows of different configurations simultaneously.

4.2. Flow over periodic hills: generalizability test

The proposed method is further used to train the neural network-based model for the flows over periodic hills. The flow with the slope of $\alpha = 1$ is used to train the model. The ensemble-based method is capable of reconstructing the flow field accurately in this case. This is shown in Figure 6 where the velocity contour is provided with comparison to the baseline $k$–$\varepsilon$ model and the DNS results. It can be seen that the flow characteristics are well captured, through minimizing the discrepancies between the prediction and the given data. It is noted that only four velocity profiles at $x/H = 1, 3, 5, 7$ are used to achieve the improved reconstruction of the entire field. The separation bubbles with the $k$–$\varepsilon$ model, the learned model, and the truth are also provided in Figure 6. Obviously, the prediction with $k$–$\varepsilon$ model significantly underestimates the separation bubble size, while the learned model well captures the bubble structure.

To clearly show the improvement in the velocity prediction, we present the comparison
Ensemble-based learning of turbulence model

$k$–$\varepsilon$ learned model DNS

Figure 6: Contour plots of the prediction results in velocity from the $k$–$\varepsilon$ model, the learned model, and DNS for the periodic hill case.

\begin{center}
\begin{tabular}{c c c}
\hline
 & DNS & $k$–$\varepsilon$ & learned model \\
\hline
$u_x$ & & & \\
$u_y$ & & & \\
$\|u\|$ & & & \\
\hline
\end{tabular}
\end{center}

Figure 7: Comparison of velocity and the friction coefficient $c_f$ at the bottom wall along profiles between the $k$–$\varepsilon$ model, the learned model, and the DNS for the periodic hill case.

along profiles in Figure 7 (a) and (b). The velocity profiles are improved significantly compared to the $k$–$\varepsilon$ model predictions. Particularly, in the separation region, both the velocity $u_x$ and $u_y$ are well predicted which is in good agreements with the DNS results. The comparison in the friction coefficient along the bottom wall is plotted in Figure 7(c). The position of the reattachment point with the $k$–$\varepsilon$ model deviates substantially from the DNS. In contrast, the learned model is capable of improving significantly the prediction of friction coefficient, and especially the reattachment point is very close to the truth.

The learned model has good predictive performance in capturing flow features for $\theta_1$ and $\theta_2$ at intermediate or small magnitudes. This is illustrated in Figure 8(a) where the comparison of scalar invariants among the baseline ($k$–$\varepsilon$ model), the learned model, and
Figure 8: Panel (a) shows contour plots of $\theta_1$, $\theta_2$, and $|\theta_1| - |\theta_2|$ with comparison among the baseline $k-\varepsilon$ model, the learned model, and the DNS; Panels (b) and (c) show kernel density plot of $\theta_1$ and $\theta_2$ from the truth and the prediction for periodic hill case, respectively. The round circles in panels (b) and (c) indicate the values of the 30% quantiles (i.e., 30% of the cells have $\theta$ larger than this value in the magnitude). The probability densities of the truth and the prediction are plotted on the margins.

The nonlinear mapping between the scalar invariant $\mathbf{\theta}$ and the $g$ function is learned from the training data. The functional mapping is shown in Figure 9. In this case no ground truth of the mapping $\mathbf{\theta} \mapsto g$ exists for validation. Here we show the baseline mapping from the linear eddy viscosity, i.e., $g^{(1)} = -0.09$ and $g^{(2)} = 0$. The function is almost constant at about $-0.098$ for $g^{(1)}$ and $0.01$ for $g^{(2)}$. The $g$ function varies slightly the DNS data are presented. The scalar invariants $\mathbf{\theta}$ from the learned model exhibit patterns similar to those of the DNS data, while the noticeable difference exists mainly in the $\mathbf{\theta}$ with large magnitudes. The comparisons of $\theta_1$ and $\theta_2$ between the prediction and the truth are presented in Figures 8(b) and 8(c), respectively. The plots of kernel densities in Figures 8(b) and 8(c) indicate that there are relatively small number of cells with input features $\theta_1$ and $\theta_2$ with large magnitudes. Specifically, only 30% of the cells in the domain have magnitudes of $\theta_1$ and $\theta_2$ larger than 5.0 and 3.8, respectively. This is the probable cause of the deteriorated prediction in the regions with large $\theta$ magnitudes (Figures 8a).
Ensemble-based learning of turbulence model

Figure 9: Plots of the learned mapping between the scalar invariants $\theta$ and the tensor coefficient $g$ with comparison to the baseline model for periodic hill case. The dash lines indicate the plane of $g^{(1)} = -0.09$ and $-0.098$ in panel (a) and $g^{(2)} = 0$ and $0.01$ in panel (b).

for the large invariant $\theta_1$ and the small invariant $\theta_2$ which is mainly in the uphill region with large strain rates.

Our generalizability test suggests that the learned model is able to generalize to cases that are similar (in terms of feature space) to the trained cases but perform less well in cases with large differences from the trained cases. This test suggests that a wide range of input features should be embedded in order to obtain a practical model. The results of the predicted velocity $u_x$ for different slopes $\alpha$ are shown in Figure 10. All the cases show that the learned model can noticeably improve the mean flow estimation in terms of the velocity compared to the baseline $k$–$\varepsilon$ model. Particularly, for the case of $\alpha = 0.8$ and $\alpha = 1.2$, the velocity profiles $u_x$ have a remarkable agreement with the DNS data. That is probably due to the similar input features of these two cases to those of the training case of $\alpha = 1$. Additionally, the error between the prediction and the DNS data over the entire field and the recirculation region ($0 < x/H < 5$ and $0 < y/H < 1$) is shown in Figures 10(e) and 10(f), respectively. It is obvious that the learned model provides better prediction than the baseline $k$–$\varepsilon$ model in all the test cases. For the training case ($\alpha = 1$), the learned model provides the lowest prediction error, which is reasonable since the prediction is directly informed by the training data. The model prediction error increases as the extrapolation case is further away from the training case. Particularly there exhibit noticeable discrepancies in the case of $\alpha = 1.5$. The maximum value of the input feature is provided in Table 4 to show the feature difference among these cases. It can be seen that the range of the input feature for $\alpha = 0.8$ and $1.2$ is relatively close to the training case in contrast to the cases of $\alpha = 0.5$ and $1.5$. This confirms that the consistency of the input features between the training case and the test cases is essential for the generalizability of the data-driven model. For the flow with similar input features the trained model is able to provide satisfactory predictions. This suggests that a wide range of input features should be included in the training case to obtain a practical model.
Geometry (slope parameter $\alpha$) $0.5$ $0.8$ $1.0$ $1.2$ $1.5$

Max. of input feature $\theta_1$ $161$ $115$ $105$ $91$ $69$

Exceeds training case ($\alpha = 1$) by $53.3\%$ $9.52\%$ $0$ $13.3\%$ $34.3\%$

Table 4: Comparison of the maximum value of input features in flow configurations with different slopes $\alpha$.

---

Figure 10: Results of generalizability tests on configurations with different slopes ($\alpha = 0.5, 0.8, 1.2, 1.5$). The panels (a)–(d) show the velocity profiles with comparison to the $k-\varepsilon$ model and the DNS. The panels (e) and (f) show the plots of prediction error over the entire field and recirculation region, respectively. The shadow in panel (f) indicate the recirculation region for error calculations.
5. Discussion

5.1. Parallelization

To enhance the generalizability of the learned model, training data should embed various flow features from different configurations, e.g., the square duct, the periodic hills, and airfoils. To handle a large data set, the conventional machine learning training algorithms need to randomly split the data into multiple batches. Further, the stochastic gradient descent (SGD) is employed to train the model by looping over the entire data set sequentially (Kovachki & Stuart 2019). This makes it inefficient for handling the large data set. The ensemble-based framework is able to learn the model from a large data set in a parallelable manner. The ensemble-based method is inherently parallelizable and can handle the data with random noise so as to avoid data overfitting. This achieves the same goal as SGD for machine learning. Furthermore, the model-consistent training framework can train with the data from different configurations simultaneously. These training cases do not need communication (e.g., embarrassingly parallel workload), such that the wall time is not increased when the number of used CPU cores is equal to the number of configurations.

5.2. Flexibility in learning from different observation data

The ensemble-based framework is extremely flexible in terms of the loss function, specific applications, and observation data, due to its derivative-free nature. Specifically, the loss function can even be non-differentiable, e.g., when learning dynamic model parameters with statistical observation data. In such a scenario, the adjoint-based method would be difficult to deploy, while the ensemble method only needs to evaluate the cost function and approximate the corresponding gradient based on the model input and output. Moreover, the framework here is used for the turbulence closure problem. Other physical systems where the adjoint solver is not readily available can apply the proposed method to learn the underlying closure model based on the measurable observations. Besides, in specific cases, e.g., the RANS modelling, the available data are often collected from different configurations with varying physical quantities and dimensionality. It is difficult for the conventional methods to use these disparate data, as they need to develop specific adjoint solvers for different measurable quantities, which is a challenging task for complex CFD solvers. The proposed model-consistent learning framework can approximate the sensitivity of the model prediction to the model parameters based on model inputs and outputs. With the non-intrusive and derivative-free nature, the ensemble-based model-consistent learning is naturally flexible for different loss functions, physical systems, and disparate data.

6. Conclusion

This work proposes an ensemble-based framework to learn nonlinear eddy viscosity turbulence models from indirect data. The proposed framework has three major advantages. First, the training is non-intrusive without an adjoint solver. Second, the framework ensures the consistency between the training and the prediction environments and thus improves the robustness of the learned model. Finally, the embedded Reynolds stress model based on a tensor basis neural network is flexible and has the potential of representing a universal or unified turbulence model. The ensemble-based method has been shown to be able to learn a turbulence model from indirect observation data more efficiently than the adjoint-based method.

The capability of the proposed framework is demonstrated on two flows, the flow...
in a square duct and the flows over periodic hills. The duct flow demonstrated the capability of the proposed method in learning underlying closure relationships from velocity observation data, and the periodic hill case showed the generalizability of the learned model to flows in different geometries. Both cases highlight the straightforward implementation of the ensemble-based learning method. It runs in parallel and can learn from large sets of training flows simultaneously. Moreover, the non-intrusive nature of the ensemble-based method makes it convenient to handle different types of observations without developing an adjoint solver for each new objective function. The proposed framework is promising for learning turbulence models in industrial applications.

Appendix A. Practical implementation

The practical implementation of the proposed ensemble-based model-consistent turbulence modelling framework is detailed in this section and illustrated schematically in Figure 11. Given the observation error $R$, the data set $y$, and the sample variance $\sigma$, the procedure for the ensemble-based model learning is summarized below:

(i) Pre-training: To obtain the initial weight $w^0$ of the neural network, we pre-train the network to be an equivalent linear eddy viscosity model such that $g^{(1)} = -0.09$ and $g^{(i)} = 0$ (for $i = 2$ to 10). The weights so obtained, $w^0$, is set as the initial value for optimization (Michelén Ströfer & Xiao 2021). The pre-training is necessary because conventional initialization methods (e.g., random initialization) may lead to nonphysical values such as the positive $g_1$ (negative eddy viscosity), which would cause divergence of the RANS solver. Pre-training is needed to address this difficulty and accelerate model learning.

(ii) Initial sampling: We assume that the weights to be independent and identically distributed (i.i.d.) Gaussian random variables with mean $w^0$ and variance $\sigma^2$. We draw random samples of the weights (Fig. 11a) through the formula $w_j = w^0 + \epsilon_j$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$.

(iii) Feature extraction: the velocity field $u$ and turbulence time scale $k$ are used to compute the scalar invariants $\theta$ and the tensor bases $T$ (Fig. 11b) based on the equations (2.5) and (2.6). The scalar invariants are then adopted as the inputs of the neural network function $g$, while the tensor bases are employed to construct the Reynolds stress by combining with the outputs of the neural network as illustrated in step (iv) below. The input features of the neural network are scaled into the range of $[0, 1]$.

(iv) Evaluation of Reynolds stress: input features $\theta$ are propagated to the basis coefficient $g$ with each realization of the weights $w$, and then the Reynolds stress can be constructed (Fig. 11c) through combining the coefficient $g$ and the tensor basis $T$, i.e., $\tau = 2k \sum_i g^{(i)} T^{(i)} + \frac{2}{3} \tau I$.

(v) Propagation to velocity: the velocity is obtained by solving the RANS equations for each constructed Reynolds stress. Moreover, the turbulence kinetic energy and the dissipation rate is obtained by solving the turbulence transport equations (Fig. 11d).

(vi) Computation of Kalman gain from samples. To this end, we first compute the
square root matrices at iteration step $l$ as follows:

$$S'_w = \frac{1}{\sqrt{N_e - 1}} \left[ w'_1 - \bar{w}', w'_2 - \bar{w}', \ldots, w'_{N_e} - \bar{w}' \right], \quad (A 1a)$$

$$S'_y = \frac{1}{\sqrt{N_e - 1}} \left[ \mathcal{H}[w'_1] - \mathcal{H}[\bar{w}'], \mathcal{H}[w'_2] - \mathcal{H}[\bar{w}'], \ldots, \mathcal{H}[w'_{N_e}] - \mathcal{H}[\bar{w}'] \right], \quad (A 1b)$$

$$\bar{w}' = \frac{1}{N_e} \sum_{j=1}^{N_e} w'_j, \quad (A 1c)$$

where $N_e$ is the sample size. The Kalman gain matrix is then computed as:

$$K = S_w S_y^\top (S_y S_y^\top + \gamma^l R)^{-1}. \quad (2.8)$$

(vii) Update weights of neural networks: use the iterative ensemble Kalman method to update the weights of the neural network (Fig. 11e), i.e.,

$$w^{l+1}_j = w^l_j + K (y_j - \mathcal{H}[w^l_j])$$

In steps (vi) and (vii), the iteration size parameter $\gamma$ is adjusted in an inner loop. Specifically, we let $\gamma^v = \beta^v \{S_y^v (S_y^v)^\top \} / \{R\}$ where $\beta^v$ is a scalar coefficient whose value changes at each subiteration index $v$. Specifically, at each iteration, an initial value (i.e., at sub-iteration step $v = 0$) is set to be $\beta^0 = 1$. If at the $v$-th sub-iteration step, the average data misfit (over the ensemble of model predictions) is reduced, then at the next sub-iteration step, we set $\beta^{v+1} = 0.8\beta^v$ and break out of the inner loop; otherwise we set $\beta^{v+1} = 1.2\beta^v$ and repeat step (vi). We allow for up to five sub-iterations in this inner loop.

(viii) If the ensemble variance is smaller than the observation error, consider it converged and end iteration; otherwise, continue to step (iii) until the convergence criterion above is met or the maximum number of iterations is reached.

Appendix B. Hessian matrix in ensemble Kalman method

In this section we illustrate how the approximated Hessian matrix as well as the gradient (Jacobian) are implicitly incorporated in the ensemble Kalman method, which leads to accelerated learning and improved robustness. This is a crucial difference compared to the stochastic gradient descent optimization used for neural network training in deep learning.

The weight update scheme of the iterative ensemble Kalman method is formulated as in Equation (2.8):

$$w^{l+1}_j = w^l_j + K (y_j - \mathcal{H}[w^l_j]) \quad \text{with} \quad K = S_w S_y^\top (S_y S_y^\top + \gamma^l R)^{-1}. \quad (2.8)$$

We first establish its connection to the common form of the Kalman gain matrix $K = PH^\top (HPH^\top + \gamma^l R)^{-1}$. To this end, we write the prediction error covariance matrix $P$ and other associated quantities in terms of the square root matrix $S_w$ and its projection $S_y$ to the observation space, i.e.,

$$P = S_w S_w^\top \quad \text{and} \quad S_y = HS_w. \quad (B 1)$$

Consequently, the cross-covariance $PH^\top$ between the weights $w$ and the predictions $\mathcal{H}[w]$ and the projection of $P$ to the observation space are:

$$PH^\top = S_w S_y^\top \quad \text{and} \quad HPH^\top = S_y S_y^\top,$$
respectively. The two forms of the Kalman gain matrix are thus established.

Next, we show that the Kalman gain matrix $K$ in the update scheme implicitly contains the inverse of an approximated Hessian matrix of the cost function. To see this point, let $H$ be the local gradient of the observation operator $\mathcal{H}$ (with respect to the parameter $\mathbf{w}$; same for all gradient and Hessian mentioned hereafter). After dropping the iteration index, it can be shown that the gradient of the data misfit term $\| \mathbf{y} - \mathcal{H}[\mathbf{w}] \|^2_{\gamma_R}$ in Equation (2.7) is given by (neglecting a constant factor 2)

$$-H^T (\gamma_R)^{-1} \left( y_j - \mathcal{H}[w_j^i] \right), \quad (B\ 2)$$

and the local Hessian matrix of the entire objective function is given by (neglecting a constant factor 2)

$$P^{-1} + H^T (\gamma_R)^{-1} H. \quad (B\ 3)$$

We will utilize the following matrix identity:

$$PH^T \left( PH^T + \gamma R \right)^{-1} = \left( P^{-1} + H^T (\gamma R)^{-1} H \right)^{-1} H^T (\gamma R)^{-1}. \quad (B\ 4)$$

See Equation (49) in Luo (2021) for detailed derivations of the identify above. In general, the observation operator $\mathcal{H}$ is nonlinear, in which case the square root matrix $S_y$ as estimated in Equation (A 1) provides a derivative-free approximation to the projected square root matrix $HS_w$. Accordingly, one can see that the term $K \left( y_j - \mathcal{H}[w_j^i] \right)$ in Equation (2.8) is an ensemble-based derivative-free approximation to the product between the inverse of the local Hessian matrix in Equation (B 3) and the (negative) local gradient
Appendix C. Sensitivity study of network architecture and observation data

Neural networks with different architectures are used in the model consistent training of the square duct case to show the sensitivity of the framework. Three network architectures are tested: (1) two hidden layers with 5 neurons per layer (baseline), (2) two hidden layers with 10 neurons per layer, and (3) ten hidden layers and 10 neurons per layer. The results of errors in the velocity and Reynolds stress over the entire field are summarized in Table 5. It can be seen that the results are not very sensitive to the neural network architecture for the square duct case. The misfit in the velocity and the prediction error in the Reynolds stress among the three cases are very similar. It is noted that the case with 2 layers and 5 neuron per layer is able to predict well the flow fields in both velocities and the Reynolds stresses. This is likely due to the narrow range of the input features in this case. The maximum of the input features is approximately 7, which can be sufficiently captured with 69 parameters in the neural network. Moreover, we test the setting of using the velocity observation along the anti-diagonal line of the computational domain. The results in both the velocity and the Reynolds stress field become slightly inferior compared to the case with the full field.

Acknowledgment

XLZ, and GH are supported by the NSFC Basic Science Center Program for “Multi-scale Problems in Nonlinear Mechanics” (No. 11988102). XLZ also acknowledges supports from the National Natural Science Foundation of China (No. 12102435) and the China Postdoctoral Science Foundation (No. 2021M690154).

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan,
Brener, B. P., Cruz, M. A., Thompson, R. L. & Anjos, R. P. 2021 Conditioning and accurate solutions of Reynolds average Navier–Stokes equations with data-driven turbulence closures. *Journal of Fluid Mechanics* 915, 1–20.

Chen, Y., Chang, H., Meng, J. & Zhang, D. 2019 Ensemble Neural Networks (ENN): A gradient-free stochastic method. *Neural Networks* 110, 170–185.

Duraisamy, K. 2021 Perspectives on machine learning-augmented Reynolds-averaged and large eddy simulation models of turbulence. *Physical Review Fluids* 6 (5), 050504.

Eisenfeld, B., Rumsey, C. & Togitti, V. 2016 Verification and validation of a second-moment-closure model. *AIAA Journal* 54 (5), 1524–1541.

Evensen, G. 2018 Analysis of iterative ensemble smoothers for solving inverse problems. *Computational Geosciences* 22 (3), 885–908.

Han, J., Zhou, X. H. & Xiao, H. 2022 VCNN-e: A vector-cloud neural network with equivariance for emulating Reynolds stress transport equations. *arXiv preprint arXiv:2201.01287*.

Holland, J. R., Baeder, J. D. & Duraisamy, K. 2019 Field inversion and machine learning with embedded neural networks: Physics-consistent neural network training. In *AIAA Aviation 2019 Forum*, p. 3200.

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

Lauder, B. E., Reece, G. J. & Rodi, W. 1975 Progress in the development of a Reynolds-stress turbulence closure. *Journal of Fluid Mechanics* 68 (3), 537–566.

Lauder, B. E. & Sandham, N. D. 2002 *Closure strategies for turbulent and transitional flows*. Cambridge University Press.

Lauder, B. E. & Sharma, B. I. 1974 Application of the energy-dissipation model of turbulence to the calculation of flow near a spinning disc. *Letters in Heat and Mass Transfer* 1 (2), 131–137.

Ling, J., Kurzawski, A. & Templeton, J. 2016 Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics* 807, 155–166.

Luo, X. 2021 Novel iterative ensemble smoothers derived from a class of generalized cost functions. *Computational Geosciences* 25 (3), 1159–1189.

Luo, X., Stordal, A. S., Lorentzen, R. J. & Nævdal, G. 2015 Iterative ensemble smoother as an approximate solution to a regularized minimum-average-cost problem: theory and applications. *SPE Journal* 20 (05), 962–982.

MacArt, J. F., Sirignano, J. & Freund, J. B. 2021 Embedded training of neural-network subgrid-scale turbulence models. *Physical Review Fluids* 6 (5), 050502.

Michelén Ströfer, C. A. & Xiao, H. 2021 End-to-end differentiable learning of turbulence models from indirect observations. *Theoretical and Applied Mechanics Letters* p. 100280.

Michelén Ströfer, C. A., Zhang, X. L. & Xiao, H. 2021a DAFI: An open-source framework for ensemble-based data assimilation and field inversion. *Communications in Computational Physics* 29 (5), 1583–1622.

Michelén Ströfer, C. A., Zhang, X.-L. & Xiao, H. 2021b Ensemble gradient for learning turbulence models from indirect observations. *Communications in Computational Physics* 30 (5), 1269–1289.

Michelén Ströfer, C. A., Zhang, X.-L., Xiao, H. & Coutier-Delgosha, O. 2020 Enforcing boundary conditions on physical fields in Bayesian inversion. *Computer Methods in Applied Mechanics and Engineering* 367, 113097.

Park, J. & Choi, H. 2021 Toward neural-network-based large eddy simulation: application to turbulent channel flow. *Journal of Fluid Mechanics* 914.

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

Pope, S. B. 2000 *Turbulent flows*. Cambridge University Press.

Schmelzer, M., Dwight, R. P. & Cinnella, P. 2020 Discovery of algebraic Reynolds-stress
models using sparse symbolic regression. *Flow, Turbulence and Combustion* **104** (2), 579–603.

Schneider, T., Stuart, A. M. & Wu, J.-L. 2020a Ensemble Kalman inversion for sparse learning of dynamical systems from time-averaged data. *arXiv preprint arXiv:2007.06175*.

Schneider, T., Stuart, A. M. & Wu, J.-L. 2020b Imposing sparsity within ensemble Kalman inversion. *arXiv preprint arXiv:2007.06175*.

Shih, T.-H. 1993 A realizable Reynolds stress algebraic equation model. , vol. 105993. National Aeronautics and Space Administration.

Slotnick, J., Khodadoust, A., Alonso, J., Darmofal, D., Gropp, W., Lurie, E. & Mavriplis, D. 2014 CFD vision 2030 study: a path to revolutionary computational aero sciences. In NASA CR-2014-218178. Langley Research Center.

Spalart, P. R. 2000 Strategies for turbulence modelling and simulations. *International Journal of Heat and Fluid Flow* **21** (3), 252–263.

Spalart, P. R. & Allmaras, S. R. 1992 A one-equation turbulence model for aerodynamic flows. AIAA Paper 1992-439.

Speziale, C. G., Sarkar, S. & Gatski, T. B. 1991 Modelling the pressure–strain correlation of turbulence: an invariant dynamical systems approach. *Journal of Fluid Mechanics* **227**, 245–272.

Sun, Luning & Wang, Jian-Xun 2020 Physics-constrained Bayesian neural network for fluid flow reconstruction with sparse and noisy data. *Theoretical and Applied Mechanics Letters* **10** (3), 161–169.

The OpenFOAM Foundation 2021 *OpenFOAM User Guide*.

Wallin, S. & Johansson, A. V. 2000 An explicit algebraic Reynolds stress model for incompressible and compressible turbulent flows. *Journal of Fluid Mechanics* **403**, 89–132.

Wang, J.-X., Wu, J.-L. & Xiao, H. 2017 Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data. *Physical Review Fluids* **2** (3), 034603.

Weatheritt, J. & Sandberg, R. 2016 A novel evolutionary algorithm applied to algebraic modifications of the RANS stress–strain relationship. *Journal of Computational Physics* **325**, 22–37.

Wu, J.-L., Michelén Ströfer, C. A. & Xiao, H. 2019a Physics-informed covariance kernel for model-form uncertainty quantification with application to turbulent flows. *Computers & Fluids* **193**, 104292.

Wu, J.-L., Xiao, H., Sun, R. & Wang, Q. 2019b Reynolds-averaged Navier–Stokes equations with explicit data-driven Reynolds stress closure can be ill-conditioned. *Journal of Fluid Mechanics* **869**, 553–586.

Xiao, H. & Cinnella, P. 2019 Quantification of model uncertainty in RANS simulations: A review. *Progress in Aerospace Sciences* **108**, 1–31.

Xiao, H., Wu, J.-L., Laizet, S. & Duan, L. 2020 Flows over periodic hills of parameterized geometries: A dataset for data-driven turbulence modeling from direct simulations. *Computers & Fluids* **200**, 104431.

Yang, X. IA & Griffin, K. P 2021 Grid-point and time-step requirements for direct numerical simulation and large-eddy simulation. *Physics of Fluids* **33** (1), 015108.

Zhang, X.-L., Xiao, H., Gomez, T. & Coutier-Delgosha, O. 2020 Evaluation of ensemble methods for quantifying uncertainties in steady-state CFD applications with small ensemble sizes. *Computers & Fluids* p. 104530.

Zhang, X.-L., Xiao, H., Luo, X. & He, G. 2022 Ensemble-based learning of turbulence models. Software available from github.com/xiaoh/DAFI/ensemble-learning.

Zhao, Y., Akolekar, H. D., Weatheritt, J., Michelassi, V. & Sandberg, R. D. 2020 RANS turbulence model development using CFD-driven machine learning. *Journal of Computational Physics* **411**, 109413.

Zhou, X. H., Han, J. & Xiao, H. 2022 Frame-independent vector-cloud neural network for nonlocal constitutive modeling on arbitrary grids. *Computer Methods in Applied Mechanics and Engineering* **388**, 114211.